MEDIATING BACTERIAL COMMUNICATION: THE ROLE OF SMALL RNA IN QUORUM SENSING

by

Geoffrey Alexander Munro Hunter

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Mathematics

The University of Utah

December 2013

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of Geoffrey Alexander Munro Hunter has been approved by the following supervisory committee members:

James P. Keener	, Chair	Oct 9, 2013
Frederick R. Adler	, Member	Oct 9, 2013
Aaron Fogelson	, Member	Oct 9, 2013
David Blair	, Member	Oct 9, 2013
Colin Dale	, Member	Oct 11, 2013

and by Peter E. Trapa, Chair of the Department of Mathematics and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

Newly discovered, noncoding RNA regulate cellular processes and gene expression. Simple model systems, such as quorum sensing systems, are studied to understand the mechanisms by which these RNA act on their targets. Quorum sensing is a process by which bacteria coordinate expression of their genes based on the local cell-population density. Up to five noncoding RNA, called small RNA (sRNA), in the quorum sensing systems of *Vibrio harveyi* and *Vibrio cholerae* regulate expression of the master transcriptional regulator, LuxR (*V. harveyi*) and HapR (*V. cholerae*). LuxR/HapR regulate genes associated with virulence and bioluminescence that are downstream of the quorum sensing system.

The *V. harveyi* and *V. cholerae* quorum sensing systems are topologically identical and their components are homologous, yet each responds differently under identical experimental conditions. Experiments show that all sRNA are necessary in *V. harveyi* and any single sRNA is sufficient in *V. cholerae* to repress bioluminescence. Hence, Qrr are additive in *V. harveyi* and redundant in *V. cholerae*. Subsequent experiments have shown that feedback in the sRNA circuit increases the expression of Qrr when one or more Qrr are removed. Differences in the tuning of this feedback are thought to cause the additive and redundant Qrr phenotypes; however, this long-standing hypothesis remains untested.

In this work, a novel model of the *V. harveyi* and *V. cholerae* sRNA circuit is formulated and parameterized to identify parametric differences underlying the phenotypic differences. This yields a single model with two different parameterizations whose behavior agrees quantitatively with a variety of empirical data from *V. harveyi* and *V. cholerae*. The model, therefore, can be used for the *in silico* design, testing, and analysis of experiments and, as such, is a utility to generate experimentally verifiable hypotheses. Analysis of the model shows that the feedback in the sRNA circuit is neither necessary nor sufficient to explain the phenotypic differences, which is in contrast to the long-standing hypothesis. Rather, the additive and redundant Qrr phenotypes are emergent phenomena and, in the case of *V. harveyi* and *V. cholerae*, reflect differences in the saturation of the protein chaperon Hfq with sRNA. Overall, this suggests that Hfq is an important modulator of sRNA-facilitated repression of target mRNA.

To Mum,
for the courage to start.

To Kelly,
for the strength to finish.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This work has been made possible with the help of the following people to whom I am very grateful for their assistance over the years.

- Thank you Professor Sivabal Sivaloganathan and Professor Giuseppe Tenti for tolerating me when I was an undergraduate student and for giving me the golden boot when I needed it.

- Thank you Professor Owen Hamill for sharing the TRPC1 data and for your insightful comments on TRPC1 channels and mechanosensitive channels.

- Thank you Professor Bonnie Basser, Associate Professor Sine Svenningsen, and Dr. Kim Tu for the data leading to the work in Chapter 3 and for your insightful comments and discussions on *V. harveyi* and *V. cholerae*. I admire the rigor, thoroughness, and poignancy of your research.

- Thank you Associate Professor Steve Poelzing for your friendship, honest feedback, and lighthearted perspective on life.

- Thank you Dr. Elizabeth (Liz) Copene for being the "Editor in Chief" of my work. You accepted my requests to review my work with grace and enthusiasm and your feedback has been invaluable. Next to my family, you have also been the biggest boost to my morale in my struggle to finish for which I cannot begin to thank you enough.

- Thank you Associate Professor Frank Sachse for welcoming me into your lab and taking me under your wing as we worked on the TRPC1 research. Although Thesis 1.0 and our collaboration came to an abrupt halt, our work together introduced me to the tools and concepts that I would later apply in Chapter 3. Consequently, our work has helped me develop a broader appreciation of the significance and purpose of this research.

- Thank you Assistant Professor Fernando Guevara Vasquez for helping salvage my dissertation and for your assistance getting it published [45, 44]. Consequently, this has led to my final defense and completion of my Ph.D.

- Thank you Professor Fred Adler, Professor Aaron Fogelson, Professor David Blair, and Associate Professor Colin Dale for your feedback and support over the years. Your feedback has helped elevate my work to a higher standard that I am proud of.

- Thank you Professor James Keener for your insight, feedback, and, most importantly, for facilitating my maturation into a scientist. I bought the famous "yellow book" for myself as a Christmas present when I was an undergraduate student never thinking that it would lead me to where I am today. I am amazed with your breadth of knowledge of Math Biology/Physiology, keen analytical eye, and your knack for finding simple solutions to complicated problems.

- Thank you Mum for being you. You were instrumental in getting me here and teaching me how to survive. Yes, I know, I'll come home soon...

- Thank you Nana/Patti/Mom for sharing your chocolate chip cookie recipe with us and for coming to Utah to spend, what would end up being, your final months. We miss you every day and wish that you could be here to celebrate this with us.

- Thank you Braedon for being the best son a father could have. You are a source of constant joy in my life and an excuse for me to act like a child in public places. The hugs, tickles, and giggles we share keep me afloat every day.

- Thank you Kelly (a.k.a. the "Sixth Advisor") for being a source of strength, compassion, and encouragement in my life. You made the choice to put your "real life" on hold when you went to graduate school and again, foolishly so, for me when we got married. You've always supported me and have given me the time and space I needed to figure things out. I hope that the next chapter in our life together is even more fulfilling than our first five years have been. I love you bunches.

CHAPTER 1

INTRODUCTION

First discovered in the luminous bacterium *Vibrio harveyi* and *Vibrio fischeri*, quorum sensing is the process by which bacteria synchronize their gene expression based on local cell-population density [115]. Quorum sensing systems are found in several different bacterial species and are, therefore, thought to be common to all bacteria. Quorum sensing systems regulate genes associated with biofilm production, toxins, cell motility, type III secretion factors, bioluminescence, and those essential for symbiosis. Bacteria are thought to benefit from coordinating expression of their genes in a population-dependent manner. Such benefits can include evading the hosts' immune response, assisting in the disseminating of the species, surviving in adverse environments, and reducing metabolic stress [51, 115, 35, 77, 119, 112, 41].

Typically, genes regulated by quorum sensing systems show a distinct on/off expression phenotype, as exemplified in Figure 1.1. The data show bioluminescence in a wild-type *V. fischeri* strain as a function of optical density, which is a measure of the cell-population density. Bioluminescence is low at low cell-population density (LCD) then is upregulated at high cell-population density (HCD). Environmental factors, such as the preferred carbon source, also regulate the onset of the response [84]. Continued research into quorum sensing systems contributes to the development of ecological controls in agriculture [88] and novel antivirulence treatments in medicine [88, 25, 21, 19, 9] in addition to the discovery of novel gene regulatory mechanisms [57, 10], understanding bacterial-host interactions [80, 20, 110, 79], and evolutionary questions that are difficult to answer with higher organisms [88].

This work focuses on the quorum sensing systems of *V. harveyi* and *Vibrio cholerae*. *V. harveyi* are distributed throughout coastal and ocean waters and are commonly found around estuary sediment waters. These bioluminescent *Vibrios* are pathogenic to marine life, including pearl oysters, finfish, and especially crustaceans like the black tiger shrimp (*Penaeus monodon*). *V. harveyi* along with *Vibrio alginolyticus*, *Vibrio parahaemolyticus*, *Vibrio anguillarum*, *Vibio vulnificus*, and *Vibrio splendidus* are the etiological agents of

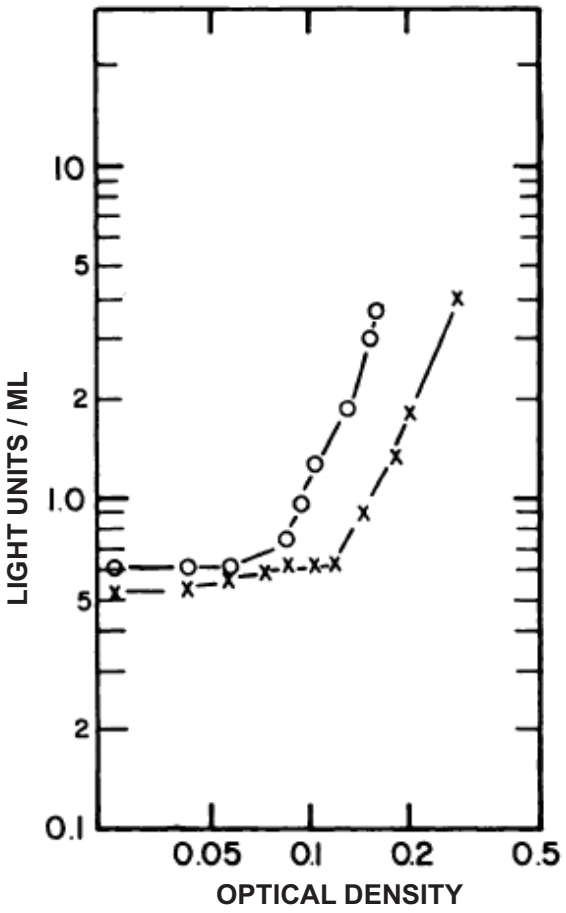

Figure 1.1. Development of bioluminescence in *V. fischeri* grown in glycerol (open circles) or glucose (X) as the energy source [84].

the epizootic disease called luminous vibriosis [101, 48, 37]. Seasonal outbreaks of luminous vibriosis can occur after heavy rainfall because the resulting nutrient runoff fuels *V. harveyi* growth. More commonly, however, *V. harveyi* is an opportunistic pathogen that targets hosts with weakened immune systems and/or those living in physiologically stressful environments (i.e., poor nutrition, cold water temperatures, confinement in a hatchery, environmental pollution) [21, 101, 72]. In 1990, an outbreak of luminous vibriosis in southeast Asia cost fisheries an estimated $1.4B (USD) alone [101]. *V. harveyi* strains are classified by different phenotypic and genetic traits, as detailed in [71, 48, 37, 109].

Although over 200 *V. harveyi* strains have been identified, only *V. harveyi* 642 and *V. harveyi* 47666-1 are known to cause luminous vibriosis. These strains acquire their virulence from a myovirus-like bacteriophage [109, 101] and inocula as low as 100 cells/ml can cause acute, devastating disease with mortality rates between 50%-100% [101, 109, 37].

Virulent *V. harveyi* strains produce hemolysin, to lyse red blood cells, metalloprotease, to degrade proteins, and lipase, to degrade lipids. Symptoms typical of luminous vibriosis in crustaceans include bioluminescence, anorexia, brittle and loose shells, brown/black spots on the shell, darkened/red body surface, tail necrosis, sparse food in the midgut, white gut disease, and, in the case of infected larvae, sluggish swimming and poor development [101, 48]. The *V. harveyi* quorum sensing system is essential for the expression and regulation of its virulence factors [22, 64, 63]. The *V. harveyi* quorum sensing system represses its virulence factors at LCD to evade the immune response of the host. Later, at HCD, the quorum sensing system coordinates expression of virulence factors so that *V. harveyi* overwhelms the hosts' immune response [35, 115, 24]. Researchers are keen to understand how pathogenicity is regulated in *V. harveyi* to curb the economic impact of luminous vibriosis.

By contrast, *V. cholerae* is found in semitropical and tropical estuarine and brackish waters and reside on phytoplankton, zooplankton, aquatic plants, crustaceans, insects, and sediments. *V. cholerae* strains are the etiological agents of the seven cholera pandemics on record since 1817. Cholera typically spreads in explosive epidemics affecting large populations and can traverse countries and continents. Early cholera pandemics have spread throughout the world and have spread to India, Bangladesh, Indonesia, Australia, Nepal, Thailand, China, USA, Brazil, and the UK. Improvements to sanitation and water treatment have substantially reduced the risk and spread of cholera. Much like *V. harveyi*, there are over 200 known *V. cholerae* strains, but only a few cause cholera pandemics. The fifth and sixth pandemics are attributed to the O1 biotype classical strain [86]. The classical strain has largely been replaced by the O1 biotype El Tor strain responsible for the current cholera pandemic that began in 1961. A new *V. cholerae* pandemic strain, *V. cholerae* O139, was identified in 1992 in India and Bangladesh and has since spread throughout southeast Asia and continues to coexist with the O1 biotype El Tor strain [32, 86]. As with *V. harveyi*, *V. cholerae* strains are also differentiated by additional phenotypic and genetic traits, as detailed in [86].

The disease progression of cholera is as follows: a host ingests undercooked seafood and/or contaminated water (i.e., by washing food with contaminated water) to become infected. Large doses of *V. cholerae* (i.e., $10^6 - 10^{11}$ CFU) are sufficient to infect healthy adults, while smaller dosages are sufficient to infect children and immunocompromised adults. After *V. cholerae* passes through the acid barrier of the stomach, they colonize the small intestine by burrowing through the mucus layer and attaching to the epithelium. At

LCD, *V. cholerae* express long filamentous fimbriae called toxin coregulated pilus (TCP) that produces a protective biofilm around the *V. cholerae* population. The TCP is also involved in the expression of other colonization factors and virulence factors [32].

TCP shares the same regulatory pathway as cholera toxin (CT), which *V. cholerae* acquires from the lysogenic filamentous bacteriophage CTXϕ. This toxin causes an efflux of chloride ions and water from the intestinal epithelial cells and leads to the severe "rice-water" diarrhea characteristic of cholera. At HCD, the *V. cholerae* quorum sensing system represses expression of TCP and CT, leading to its detachment from the intestinal epithelium and exit from the host. If left untreated, 50% of cholera cases are fatal from complications associated with acute dehydration. In the most severe cases, a host can loose up to 90 liters of fluid over a period of three days. Treatment of cholera involves electrolyte and hydration replacement therapy using oral and/or intravenous fluids. With proper treatment, the fatality rate is reduced to 1-3% of which children and infants still make up the majority of fatalities. Although other *V. cholerae* strains have been identified, only those carrying TCP and CT are associated with cholera pandemics [32, 86]. The *V. cholerae* quorum sensing system regulates expression of its virulence factors and is essential for the pathogenicity of the species. Continued research in this area may help develop novel quorum sensing inhibition therapies and *V. cholerae* vaccinations in the interest of human health [32].

1.1 Overview of the *V. harveyi* and *V. cholerae* Quorum Sensing System

V. harveyi and *V. cholerae* have similar quorum sensing systems that regulate their respective virulence factors (see Figure 1.2). Each quorum sensing system is comprised of two distinct pathways: a phosphorelay cascade that integrates cell-population density information and a sRNA circuit that regulates expression of a transcriptional regulatory protein called LuxR in *V. harveyi* and HapR in *V. cholerae* [112]. LuxR/HapR, in turn, regulate expression of all genes downstream of the quorum sensing system, including those associated with virilescence. In *V. harveyi*, three distinct autoinducers (HAI-1, AI-2, and CAI-1) are synthesized at some basal level by enzymes called autoinducer synthases (LuxM, LuxS, CqsA). For each autoinducer, there is a corresponding membrane bound receptor to which it binds: LuxN (binds HAI-1), LuxPQ (binds AI-2), and CqsS (binds CAI-1) [41, 77, 78, 114]. The autoinducers freely diffuse through the cell membrane [42] and disperse into the local environment, leaving the receptors unbound at LCD. When unbound, the receptors function as kinases and dephosphorylate high energy phosphate molecules. The phosphate is transferred to LuxU, a phosohorelay protein, that, again, transfers the phosphate to LuxO

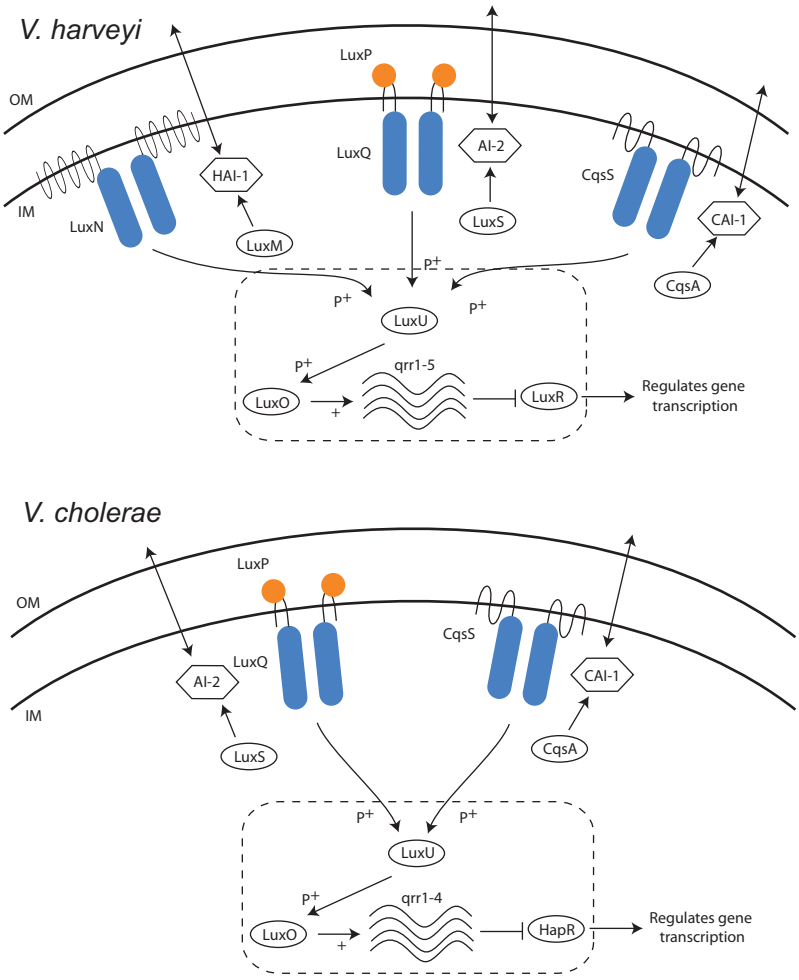

Figure 1.2. An overview of the topology and function of *V. harveyi* (top) and *V. cholerae* (bottom) quorum sensing systems at LCD. In *V. harveyi*, three phosphorelay cascades work in parallel to control the ratio of LuxO to LuxO-P based on local cell-population density. Five sRNA, *qrr1-5*, then regulate expression of the quorum sensing target genes, including the master transcriptional regulator LuxR, which upregulates downstream virulence factors. The *V. cholerae* quorum sensing circuit is nearly identical to *V. harveyi* except that *V. cholerae* has only two phosphorelay cascades, four sRNA, and its master transcriptional regulator is called HapR. The components in the *V. cholerae* quorum sensing system are homologous to those in *V. harveyi*. IM: Inner membrane, OM: Outer membrane, P^+: Phosphate.

[77]. LuxO-P activates transcription of five distinct sRNA called quorum regulated RNA (*qrr1-5*). Qrr regulate the expression of *luxR* posttranscriptionally by binding the *luxR* mRNA to prevent its translation [57, 102, 10, 103]. Therefore, LuxR/HapR is repressed because *qrr* are abundant at LCD.

Conversely, at HCD intercellular autoinducer concentration rises, leading the autoinducers to bind their respective receptors [93, 115, 75]. When bound, the receptors undergo

a conformational change that changes their function to a phosphatase [76, 113]. In this state, the flow of the phosphates is reversed as the receptors dephosphorylate LuxU, which decreases LuxO-P and *qrr*. Therefore, at HCD, LuxR/HapR is derepressed because the *qrr* concentration is low [77, 73, 52, 91].

The quorum sensing system of *V. cholerae* is nearly identical to that of *V. harveyi* with a few minor topological differences whose effects are assumed negligible. *V. cholerae* has four Qrr (*qrr1-4*) and two autoinducer receptors (LuxPQ and CqsS) rather than, respectively, the five and three found in *V. harveyi*. Experiments show that *qrr5* in *V. harveyi* is not quorum regulated [104, 102], so *qrr*5 is ignored in this work. The additional autoinducer receptor in *V. harveyi* means that *V. harveyi* responds to three, rather than two, autoinducers. However, information from the receptors is integrated into one signal – the ratio of LuxO-P to LuxO [62, 45], so the number of different phosphorelay cascades cannot be distinguished for a given ratio of LuxO to LuxO-P alone. Furthermore, the components of the *V. cholerae* quorum sensing system are homologous to those of *V. harveyi* and AI-2 and CAI-1 have the same chemical structure in both species. This means that *V. cholerae* responds to AI-2 and CAI-1 taken from *V. harveyi* and vice versa. Consequently, the nomenclature of the components in each circuit is the same between *V. harveyi* and *V. cholerae* with the exception of LuxR and HapR [42, 77].

1.1.1 The *V. harveyi* and *V. cholerae* sRNA Circuit

The sRNA circuit is central to the *V. harveyi* and *V. cholerae* quorum sensing system (see Figure 1.3). Small RNA are short fragments of noncoding RNA that regulate gene expression posttranscriptionally [10]. Qrr repress mRNA expression by binding the ribosomal binding site of target mRNA that, thereby, prevents its translation [57, 103]. The *V. harveyi* and *V. cholerae* Qrr are highly conserved within and between each species, including an identical 32bp sequence responsible for its association with mRNA [57]. At the start of the sRNA circuit, *qrr* expression is regulated by the ratio of LuxO-P to LuxO [62]. LuxO-P binds the *qrr* promoter to activate its expression. Each Qrr is rapidly degraded unless they bind Hfq [8], a protein chaperon, which also aids *qrr* to bind target mRNA. The pairing of *qrr* with mRNA results in their mutual degradation and leaves Hfq unchanged [57].

There are four regulatory pathways in the sRNA circuit to maintain precise control of *luxR/hapR* expression [77]. The first two pathways are autoregulatory loops. LuxR/HapR regulates its own expression by forming as a dimer and binding its own promoter to limit its transcription [15, 61]. Similarly, LuxO and *qrr1* are divergently transcribed, so LuxO-P limits *luxO* transcription when it binds the *qrr1* promoter. Although only LuxO-P activates

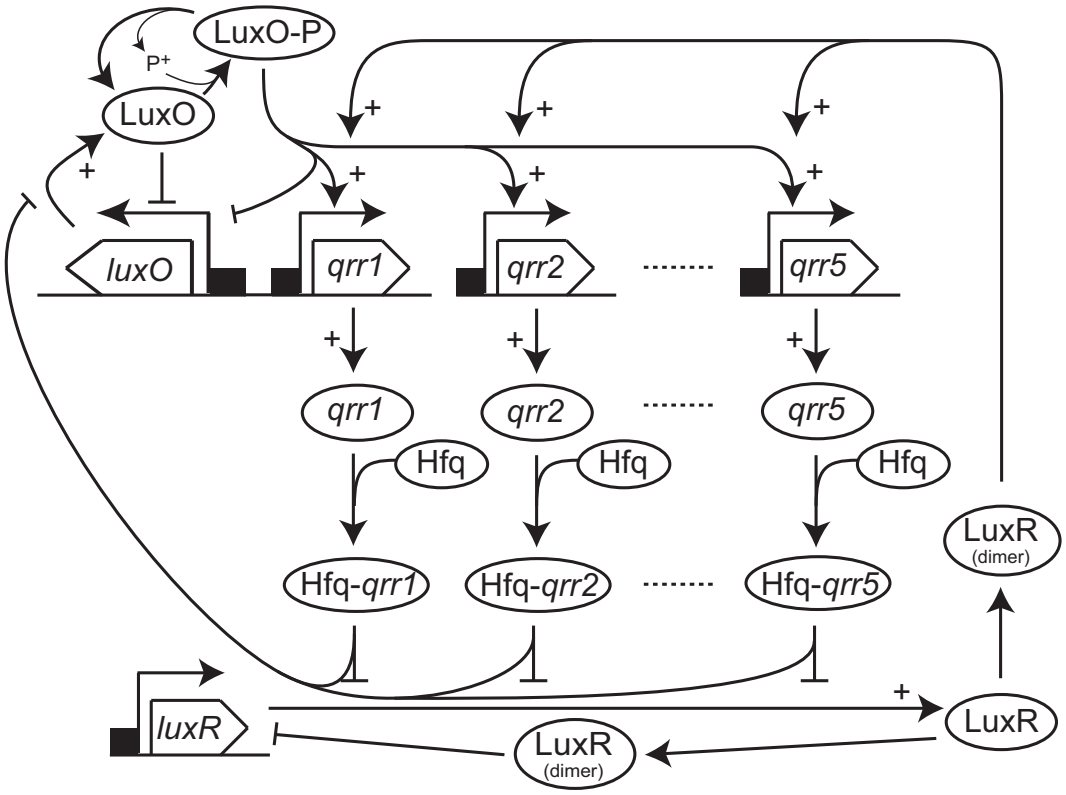

Figure 1.3. Overview of the *V. harveyi* sRNA circuit. LuxO-P activates *qrr* expression, which bind to target mRNA via Hfq to prevent translation of the mRNA into an active protein. Four different regulatory mechanisms aid to control precisely the expression of target mRNA. LuxR (as a dimer) and LuxO are autoregulatory as each binds their own promoter to limit transcription. LuxR (as a dimer) enhances *qrr* expression by binding the *qrr* promoter via the LuxR-Qrr feedback. Lastly, *qrr* target and prevent translation of *luxO* mRNA via the LuxO-Qrr feedback. The *V. cholerae* sRNA circuit is topologically identical and homologous to *V. harveyi* except that *V. cholerae* has four, rather than five, sRNA and HapR is the LuxR homologue.

qrr1 transcription, experiments show that both LuxO-P and LuxO equally inhibit *luxO* expression [103].

The remaining two pathways involve feedback between Qrr and the target mRNA and, as such, are called the LuxR/HapR-Qrr and LuxO-Qrr feedback. LuxR/HapR enhances the expression of *qrr* when LuxO-P is present. This is done by LuxR binding directly to the *qrr* promoter, while HapR does so indirectly via a currently unknown intermediary [95, 104]. Lastly, Qrr regulate LuxO expression in the same manner as Qrr regulate LuxR/HapR expression and is called the LuxO-Qrr feedback [94, 103]. These autoregulatory and feedback regulatory pathways control the onset and transition to/from LCD and HCD [94, 103].

1.2 Motivation

Although the *V. harveyi* and *V. cholerae* quorum sensing systems have homologous components and are topology identical, they respond differently to changes in Qrr. The data in Figure 1.4 show bioluminescence in a wild-type strain, isogenic strains with one Qrr only, and an isogenic strain with no Qrr for *V. harveyi* (top) and *V. cholerae* (bottom). The strains are grown over night then diluted in the morning. The dilution corresponds to a transition to a LCD environment and the initial decrease in bioluminescence in strains with at least one Qrr. At some critical cell-population density thereafter ($\sim 10^{-1}$ OD in *V. harveyi* and $\sim 10^{0}$ OD in *V. cholerae*), bioluminescence beings to increase as the bacteria enter HCD mode once again. Assuming that bioluminescence is proportional to the concentration of LuxR/HapR, the data show that all Qrr are needed to repress LuxR in *V. harveyi*, whereas any Qrr is sufficient to repress bioluminescence in *V. cholerae*. Therefore, *V. harveyi* Qrr are *additive* and *V. cholerae* Qrr are *redundant*.

Svenninsen et al. showed that Qrr feedback is responsible for increasing the expression of Qrr when one or more Qrr are removed and called this phenomenon *dosage compensation*. They then proposed that the additive and redundant Qrr phenotypes arise from differences in the tuning of dosage compensation between *V. harveyi* and *V. cholerae* [94]. To see why, consider the four simplified representations of the sRNA circuit illustrated in Figure 1.5. When *qrr1* is removed in a strain without Qrr feedback, LuxR/HapR increases and *qrr2* remains the same. This means that all Qrr are needed to repress LuxR/HapR to wild-type-like levels and, hence, the Qrr are additive. However, when *qrr1* is removed in a strain with the LuxR/HapR-Qrr feedback, *qrr2* increases via the LuxR/HapR-Qrr feedback in response to the increase in LuxR/HapR. If the LuxR/HapR-Qrr feedback is strong enough, then the increase in *qrr2* can offset the increase in LuxR/HapR. Hence, Qrr redundancy can follow from the LuxR/HapR-Qrr feedback.

Similarly, if a strain has the LuxO-Qrr feedback, then removing *qrr1* derepresses LuxR/HapR as before. However, in this case, LuxO-P increases via the LuxO-Qrr feedback to then upregulate *qrr2*. This offsets the increase in LuxR/HapR, so LuxR/HapR remains relatively unchanged provided the increase in *qrr2* is sufficient. Hence, Qrr can be redundant if the LuxO-Qrr feedback is tuned appropriately to increase *qrr2* and offset increase in LuxR/HapR. Lastly, both types of Qrr feedback work together in a wild-type strain to upregulate *qrr2* when *qrr1* is removed. If tuned appropriately, the increase in *qrr2* can offset the increase in LuxR/HapR and lead to the redundant phenotype.

This research sets out to identify the parametric differences that underly the phenotypic

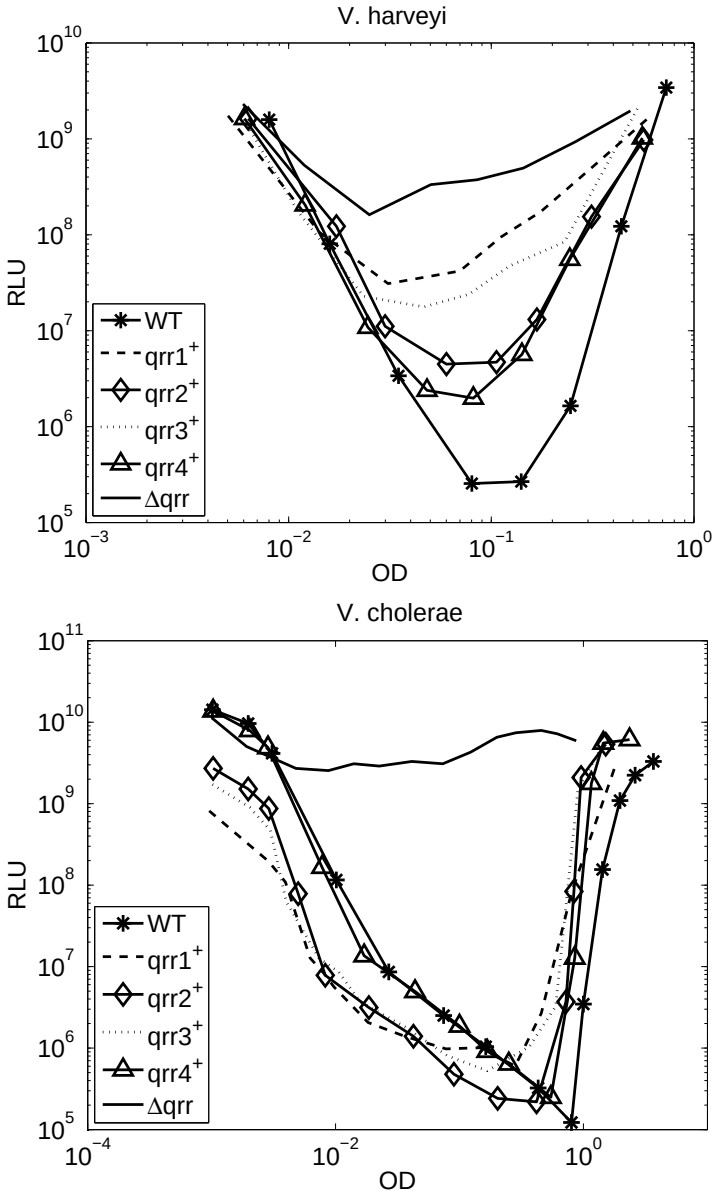

Figure 1.4. Comparison of the bioluminescence in isogenic *V. harveyi* (top) and *V. cholerae* (bottom) strains with all (wild-type), only one, or no Qrr. WT: wild-type strain, $qrr1^+$ isogenic strain with $qrr1$ only, $qrr2^+$ isogenic strain with $qrr2$ only, $qrr3^+$ isogenic strain with $qrr3$ only, $qrr4^+$ isogenic strain with $qrr4$ only, Δqrr isogenic strain with no Qrr.

differences and the specific differences that are responsible for the additive and redundant Qrr phenotypes in *V. harveyi* and *V. cholerae*. Chapter 2 begins with a review of the quorum sensing literature and highlights the contribution that mathematical models have made in the field. The chapter closes with a formulation of a novel model of the sRNA circuit in *V. harveyi* and *V. cholerae*. Chapter 3 features the parameterization of this model and shows that its behavior quantitatively agrees with a variety of data from *V. harveyi* and

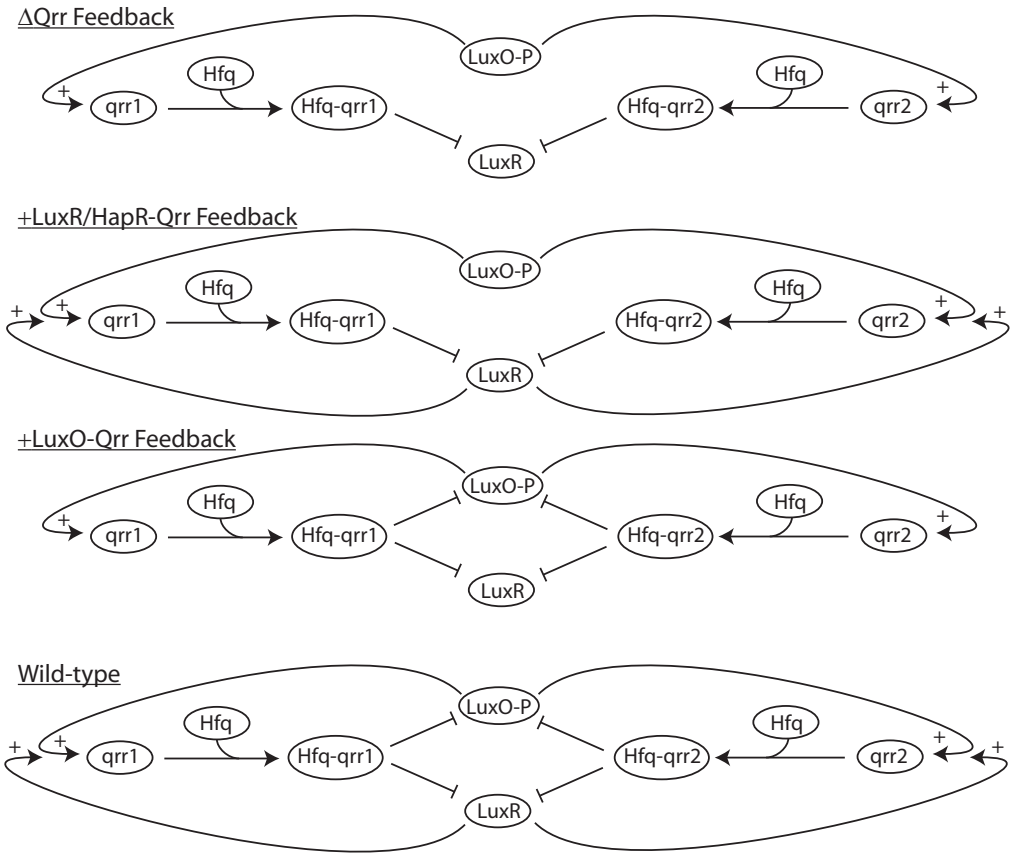

Figure 1.5. A simplified model of the *V. harveyi* and *V. cholerae* sRNA circuit to show how the Qrr feedback leads to dosage compensation and how Qrr redundancy can arise if the Qrr feedback is tuned appropriately.

V. cholerae. This also demonstrates that the model and its parameter estimates can be used for the *in silico* design, testing, and analysis of experiments in *V. harveyi* and *V. cholerae*. In Chapter 4, a general analysis of the model shows that there are up to 30 different combinations of parametric constraints that each can lead to Qrr redundancy. Using the parameter estimates of *V. harveyi* and *V. cholerae* from Chapter 3, a set of three constraints is identified as those underlying the Qrr phenotypes in *V. harveyi* and *V. cholerae*. Contrary to the dosage compensation hypothesis, Qrr feedback and, hence, dosage compensation is neither necessary nor sufficient to explain the additive and redundant Qrr phenotypes and dosage compensation diminishes the more redundant the Qrr. This means that the additive and redundant Qrr phenotypes is an emergent phenomenon in general and, in the case of *V. harveyi* and *V. cholerae*, reflects differences in the total concentration of Hfq-Qrr. Lastly, an experiment is performed *in silico* to test both hypotheses and serves as a method to validate the results.

CHAPTER 2

BACKGROUND

This chapter reviews the ecology and biology of quorum sensing systems and the contribution that mathematical models have made in the field. The chapter closes with the formulation of a novel model of the *V. harveyi* and *V. cholerae* sRNA circuit that is used for the remainder of this work.

2.1 Fundamental Assumptions of Quorum Sensing Systems

Most of the quorum sensing research over the last 30 years has focussed on understanding the genetics of quorum sensing systems and little attention has been given to understanding the evolutionary stability and/or ecological benefits of these systems. Consequently, there are three fundamental assumptions related to the benefits of quorum sensing. First, quorum sensing is assumed to be a social trait that is performed by individual cells for the good of the group. Second, quorum sensing is assumed to be beneficial at high cell-population densities. Lastly, quorum sensing is assumed to represent signalling between individuals. These assumptions have not been tested extensively either empirically or theoretically with mathematical models. This section is a summary of the main ideas and evidence in support of these assumptions featured in [88, 23].

If quorum sensing systems regulate social traits, then populations of cells that produce exofactors should be more abundant than those that do not (termed "cheats" in the evolution literature). Experiments have shown that *Pseudomonas aeruginosa* cheats are more abundant when mixed with producing *P. aeruginosa* cells and that producing *P. aeruginosa* cells are more abundant when grown in the absence of cheats. Furthermore, natural *P. aeruginosa* cheats are found in natural environments. Hence, some empirical studies with *P. aeruginosa* show that its quorum sensing system regulates social traits.

One requirement related to the sociality problem is identifying the mechanisms that contribute to its evolutionary stability. The costs of producing the exofactors must be

outweighed by their purported benefits and there should be mechanisms in place that limits the proliferation of cheats. With this in mind, some of the direct benefits associated with quorum sensing in *P. aeruginosa* also function to limit cheating. For example, the cooperative behaviors of *P. aeruginosa* are coregulated with individual fitness benefits, which is called a *pleiotropic constraint*. Hence, adopting a cheating strategy by means of a mutation may impede other activities that are beneficial to the fitness of the cell. On the other hand, *metabolic prudence* is when expression of quorum sensing factors is initiated only when the metabolic barrier associated with producing these factors is small. Another means to decrease the metabolic costs of quorum sensing is to synthesize durable products. One indirect benefit of quorum sensing systems in support of the evolutionary stability of such systems is *kin selection*. This is where the quorum sensing systems acts to enhance the proliferation of genetically similar bacteria. These studies have mainly focused on quorum sensing in *P. aeruginosa*, so there might be additional/different benefits in other species.

Quorum sensing systems are assumed to be beneficial by altering expression of certain genes at HCD, but few empirical studies test this. Of note, however, experiments with *P. aeruginosa* strains showed that its quorum sensing products are used more efficiently at HCD. Similarly, a synthetic quorum sensing system engineered in *Escherichia coli* showed that the production of quorum sensing products was beneficial at HCD and that the optimal benefit only occurs if quorum sensing is initiated at sufficiently high cell-population density. Complicating this issue, however, is the fact that a quorum sensing system can be activated in a single bacterium if it is restricted to live in a small, finite volume and/or if a bolus of autoinducer is added to the local environment. Hence, quorum sensing systems can be activated in a single bacterium and in dense colonies consisting of more than 10^6 bacteria.

2.2 Mechanisms That Regulate Gene Expression

Quorum sensing is a process that regulates expression of genes based on the local cell-population density. As such, quorum sensing systems often employ many different gene regulatory mechanisms to perform this function. These mechanisms enable bacteria to adapt to and make decisions about changes in their environment. For example, subunits of RNA polymerase called sigma factors are responsible for promoter identification and DNA binding. There are different sigma factors in bacteria and each responds to different environmental cues (i.e., pH, carbon source, etc.). When activated, the sigma factors allow RNA polymerase to bind different promoters and express different genes [59].

Bacteria also regulate their genes with transcriptional regulators. These are proteins that

act at/near the promoter or at regions farther up/downstream of the promoter. For example, transcriptional regulators may increase the affinity of RNA polymerase to the promoter and/or unwind DNA thereby to expose the promoter. Similarly, transcriptional regulators downregulate genes by blocking the promoter from RNA polymerase or by blocking the promoter by twisting DNA up/downstream of the promoter [59, 82]. Transcriptional regulators can simultaneously upregulate one gene as they downregulate a different gene [87, 70] provided the genes are divergently transcribed and share the same promoter. In this case, activation of one gene blocks the initiation site of the neighboring gene [103, 90, 26].

Newly discovered, small RNA (sRNA) also regulate genes. The sRNA are short (50-250 nucleotides long), noncoding fragments of RNA with a brief half-life. The sRNA regulate genes posttransciptionally by binding target mRNA. This changes the affinity of the ribosome to its binding site on the mRNA and/or changes the degradation rate of mRNA. Many different bacteria such as *P. aeruginosa*, *V. harveyi*, and *V. cholerae* use sRNA to regulate genes in their quorum sensing systems [10, 12, 111]. In *V. harveyi* and *V. cholerae*, sRNA repress LuxR/HapR and LuxO by preventing translation of its mRNA; however, sRNA are known to upregulate genes as well [10, 34]. In view of their structure and inherent instability, sRNA are thought to regulate genes in a fast [28, 67, 58], robust [5, 66, 60, 67, 58], and metabolically cheap [66, 69] manner. Research into the mechanisms by which sRNA regulate genes, the environmental factors that modulate this process, the ecological/evolutionary benefits of sRNA vs proteins, and identification of the genetic targets of sRNA regulation remains an active area of research.

2.3 Overview of Canonical Quorum Sensing Circuits

Although quorum sensing differs between species, all known quorum sensing systems are composed of functionally similar building blocks. First, quorum sensing systems use a diffusible chemical signal called autoinducer as a measure of cell-population density. Second, the autoinducer then binds receptors in the bacteria and the ratio of bound to unbound receptors determines the expression of a master transcriptional regulator protein. Lastly, the master transcriptional regulator is then responsible for regulating expression of the downstream quorum sensing genes.

Every known quorum sensing system is comprised of a combination of three different canonical quorum sensing circuits. The circuits are classified based on the chemical structure of the autoinducer used and how expression of the master transcriptional regulatory protein is regulated. The first quorum sensing circuit, called the LuxIR-type circuit, is found in

gram-negative bacteria. These circuits function similar to the LuxIR quorum sensing system in *V. fischeri* and is where its name is adopted from. The autoinducer in LuxIR-type circuits are acyl-homoserine-lactone based (AHL) and the synthesis of autoinducer is governed by a nonlinear positive feedback loop.

The second quorum sensing circuit is called a two-component-type circuit and is found in gram-positive bacteria. These circuits use a modified oligopeptide for an autoinducer [41] that can bind membrane bound receptors. The receptors function as kinases when unbound from autoinducer, then undergo a change in conformation to function as phosphatases when bound with autoinducer. The conformation of the receptors directs the flow of phosphates in the circuit that then regulates expression of the master transcriptional regulator. The last quorum sensing circuit, called a hybrid quorum sensing circuit, is one that uses a phosphorelay cascade with AHL-based autoinducers. These circuits function similar to two-component-type circuits rather than LuxIR-type circuits. Hybrid circuits are found in many *Vibrio* species, including *V. harveyi* and *V. cholerae* [68, 41].

Every known quorum sensing system is formed from various combinations, multiples, and arrangements of the three canonical quorum sensing circuits. For example, *V. cholerae* and *V. harveyi* have, respectively, two and three hybrid circuits arranged in parallel [68, 62]. *V. fischeri* has one hybrid circuit with two phosphorelay cascades arranged in parallel that regulate a downstream LuxIR-type circuit [68]. More complex quorum sensing systems consisting of multiple LuxIR-type circuits are also found in *P. aeruginosa*, *Burkholderia pseudomallei*, and *Burkholeria mallei* [117, 106, 105, 108, 27].

Canonical LuxIR-type circuits were the first quorum sensing circuits identified and, as such, these quorum sensing systems have directed much of the research over the last 30 years. In what follows, the basic architecture, mathematical formulation of the LuxIR-type circuit, and the contribution of these models is reviewed.

2.4 Quorum Sensing in Canonical LuxIR-Type Circuits

A canonical LuxIR-type circuit is shown in Figure 2.1 and is named after the LuxI and LuxR proteins in the *V. fischeri* quorum sensing system, which is the organism in which the circuit was first identified [74]. LuxIR-type circuits are also found in other bacterial species as detailed in Table 1 of [118]. In what follows, a mathematical model of the LuxIR-type circuit is formulated based on the work of [47, 24] then the contribution that these models have made to the field is summarized.

LuxR, R, is the master transcriptional regulator in LuxIR-type circuits. LuxR is

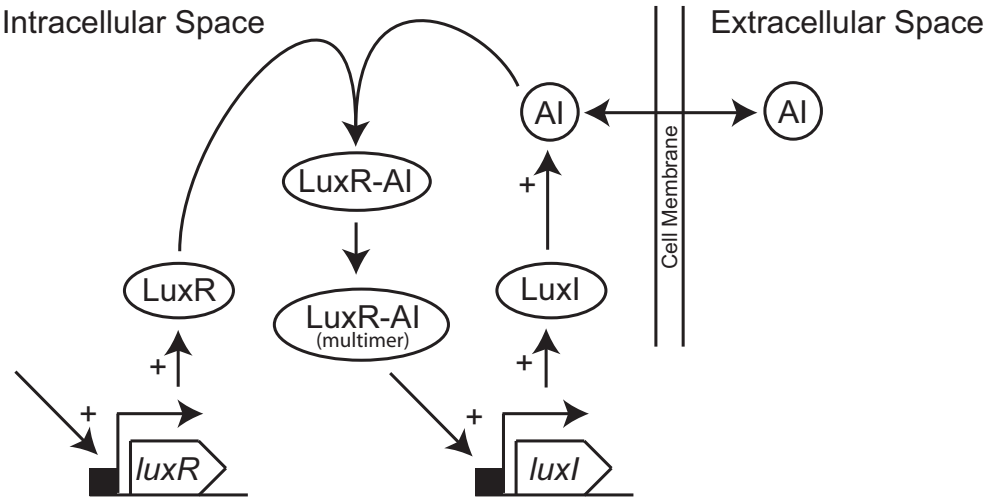

Figure 2.1. The canonical quorum sensing LuxIR-type circuit. The *luxR* and *luxI* genes are transcribed at some basal level. The LuxI protein synthesizes an AHL autoinducer (AI) that freely diffuses through the cell membrane. At LCD, autoinducer diffuses away from the colony. At high cell densities, intracellular autoinducer concentration increases, binds LuxR, and upregulates *luxI* transcription.

assumed to be transcribed at some basal rate and degraded at a rate proportional to its concentration, i.e.,

$$\xrightarrow{\kappa_R} \quad R \quad \xrightarrow{\delta_R}. \tag{2.1}$$

At HCD, the intracellular concentration of autoinducer increases and binds LuxR and forms the subunits that lead to the formation of a multimeric protein of length n [118, 35, 115].

$$nR + nA \quad \underset{\beta_{-c}}{\overset{\beta_c}{\rightleftarrows}} \quad C \tag{2.2}$$

The synthesis of autoinducer is proportional to the expression of *luxI*. For simplicity, however, we ignore *luxI* and assume that the LuxR-AI multimer synthesizes autoinducer directly, i.e.,

$$\xrightarrow{\kappa_A(C)} \quad A \quad \xrightarrow{\delta_A}, \tag{2.3}$$

where

$$\kappa_A(C) \quad = \quad \frac{V_0 + V_c K_c C}{1 + K_c C}.$$

The equations corresponding to reactions (2.1)–(2.3) describing the synthesis of autoinducer in a single cell are:

$$\frac{\mathrm{d}R}{\mathrm{d}t} \quad = \quad \kappa_R - \frac{\mathrm{d}C}{\mathrm{d}t} - \delta_R R, \tag{2.4}$$

$$\frac{\mathrm{d}C}{\mathrm{d}t} = \beta_c(RA)^n - \beta_{-c}C, \tag{2.5}$$

$$\frac{\mathrm{d}A}{\mathrm{d}t} = \frac{V_0 + V_c K_c C}{1 + K_c C} - \frac{\mathrm{d}C}{\mathrm{d}t} - \delta_A A - \phi(A - E). \tag{2.6}$$

To incorporate local cell-population density in the model, let E be the total concentration of autoinducer in the extracellular space and assume that its decay is the same as that for A, i.e.,

$$E \xrightarrow{\delta_A} . \tag{2.7}$$

Assuming that autoinducer freely diffuses across the cell membrane, the per cell flux of autoinducer into the extracellular space is proportional to the difference in autoinducer concentration across the membrane, i.e., $-\phi(A - E)$ where ϕ is the conductance. The flux term is scaled by the cell-population density, ρ, to compensate for the difference between the total concentration of autoinducer in the extracellular space vs. the total concentration of autoinducer in a cell [24]. Hence, the equation for E is

$$(1 - \rho)\left(\frac{\mathrm{d}E}{\mathrm{d}t} + \delta_A E\right) = \rho\phi(A - E), \tag{2.8}$$

where $0 \leq \rho \leq 1$ is the cell-population density.

2.4.1 Canonical LuxIR-type Circuits Are Bistable

The nonlinear, positive feedback loop the LuxIR-type circuit facilitates leads to a hysteresis loop at intermediate cell densities. To understand the stability of the model, its nondimensionalization is as follows. The following are characteristic concentrations:

$$R_0 = \frac{\kappa_R}{\delta_R}, \qquad A_0 = \frac{V_0}{\delta_A}, \qquad C_0 = \frac{\beta_c}{\beta_{-c}}(R_0 A_0)^n. \tag{2.9}$$

Then define the following dimensionless variables:

$$R = R_0\widehat{R}, \qquad A = A_0\widehat{A}, \qquad E = A_0\widehat{E}, \qquad C = C_0\widehat{C}, \qquad t = \tau\widehat{t}, \tag{2.10}$$

where τ is some characteristic time (i.e., cell growth rate). Next, define the following dimensionless parameters:

$$\begin{aligned} &\tau_R = \tau\delta_R, \qquad \tau_A = \tau\delta_A, \qquad \tau_C = \tau\beta_{-c}, \qquad \widehat{\phi} = \frac{\phi}{\delta_A}, \\ &\widehat{K}_c = K_c C_0, \qquad C_{CR} = \frac{C_0}{R_0}, \qquad C_{CA} = \frac{C_0}{A_0}, \qquad V = \frac{V_c}{V_0}. \end{aligned} \tag{2.11}$$

Lastly, 2.8–2.6 can be reexpressed in terms of the dimensionless parameters and variables to get (omitting the " $\widehat{\ }$ " notation for simplicity):

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \tau_R(1 - R) - C_{CR}\frac{\mathrm{d}C}{\mathrm{d}t}, \tag{2.12}$$

$$\frac{\mathrm{d}C}{\mathrm{d}t} = \tau_C\left((RA)^n - C\right), \tag{2.13}$$

$$\frac{\mathrm{d}A}{\mathrm{d}t} = \tau_A\left(\frac{1+VK_cC}{1+K_cC} - A - \phi(A-E)\right) - C_{CA}\frac{\mathrm{d}C}{\mathrm{d}t}, \tag{2.14}$$

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \tau_A\left(-E + \phi\frac{\rho}{1-\rho}(A-E)\right). \tag{2.15}$$

The steady state solutions for R, C and E are:

$$R = 1, \qquad C = A^n, \qquad E = (1+\phi-\Phi(\rho))A, \tag{2.16}$$

where

$$\Phi(\rho) \equiv 1 + \phi - \frac{\phi\frac{\rho}{1-\rho}}{1+\phi\frac{\rho}{1-\rho}}. \tag{2.17}$$

The equation describing the steady state solution for A is

$$0 = \frac{1+VK_cA^n}{1+K_cA^n} - \Phi(\rho)A. \tag{2.18}$$

In general, there are either one or three positive real solutions to (2.18) depending on the parameters. For example, if $n = 1$ or in the absence of positive feedback (i.e., $K_c = 0$), there is only one real solution. Therefore, the nonlinear positive feedback loop is necessary (but not sufficient) for bistability. Figure 2.2 shows a bifurcation diagram for the steady state solution of A in terms of the bifurcation parameter ρ.

Based on the above analysis, quorum sensing in canonical LuxIR-type circuits occurs as follows. At LCD, autoinducer is synthesized at some basal rate and diffuses out of the cell, so the concentration of intercellular autoinducer remains low and, hence, LuxR unbound. As cell-population density increases, so too does the intracellular concentration of autoinducer. This corresponds to moving to the right along the lower stable branch of the bifurcation diagram in Figure 2.2. When the cell-population density is sufficiently high (i.e., at $\rho \approx 0.45$ in Figure 2.2), the intracellular concentration of autoinducer binds LuxR and activates the positive feedback loop. This corresponds to a saddle node bifurcation and a sudden increase (in a steady state sense) in the concentration of autoinducer onto the upper stable branch in Figure 2.2. The concentration of autoinducer continues to increase incrementally (as a function of ρ) thereafter.

Conversely, once in the HCD state and the cell density decreases, the hysteresis loop indicates that the nonlinear positive feedback loop maintains a high rate of synthesis of autoinducer at lower cell-population densities than if the system were in its LCD state. Eventually, however, the exodus of autoinducer from the cell is greater than the concentration of autoinducer necessary to maintain the positive feedback loop. When this

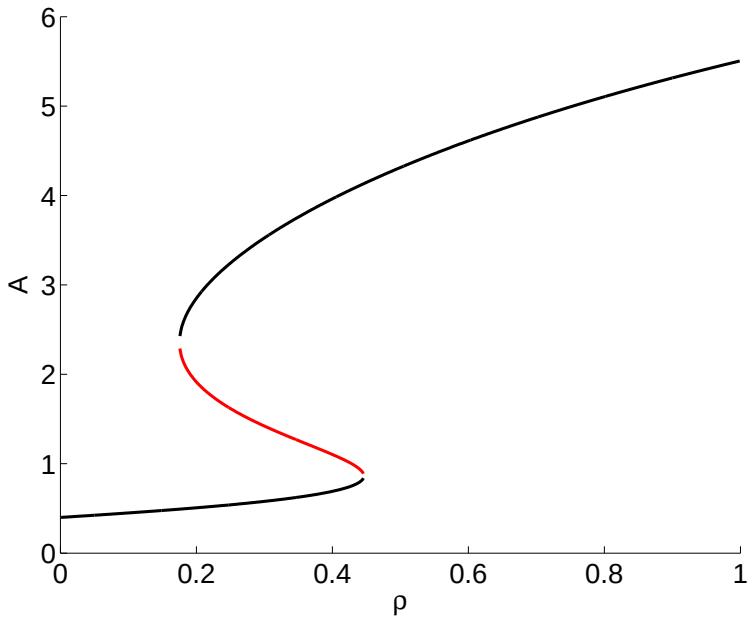

Figure 2.2. Bifurcation diagram of the canonical LuxIR-type circuit. Black and red curves respectively represent stable and unstable fixed points. Parameters are manually selected to be $n = 2, K_c = 0.127, \phi = 2.2$, and $V = 15$ (ρ is cell-population density, A is autoinducer concentration).

occurs (i.e., $\rho \approx 0.18$), autoinducer synthesis diminishes rapidly (in a steady state sense) and the system returns to its LCD state. Importantly, although this analysis shows that LuxIR-type circuits can be bistable, the existence of the hysteresis loop depends on the model parameters. Experiments have confirmed, however, that the components of the *V. fischeri* LuxIR-type quorum sensing circuit can exhibit hysteresis [116]. Several authors have noted both the importance of having and the difficulties in obtaining reliable estimates of the model parameters of LuxIR-type circuits. Furthermore, hysteresis has not been observed even in well-studied species such as *P. aeruginosa* [3, 11, 16, 39, 40, 83].

2.4.2 The Contribution of LuxIR-type Mathematical Models

Most mathematical models of quorum sensing systems in the literature represent the LuxIR-type circuits in *V. fischeri*, *P. aeruginosa*, *Agrobacterium tumefaciens*, *Staphylococcus aureus*, and *Burkholderia spp.* This might simply reflect the proportion of empirical studies available to develop such models and/or reflect the potential contribution of these models in human health. The simplest models of LuxIR-type circuits suggested that quorum sensing is a bistable switch [47, 24, 11]. These models, in turn, have laid the foundation for *in silico* design, validation, and analysis of biological networks and experiments. For

example, LuxIR-type models have been extended to understand how environmental factors such as pH, carbon source, and oxygen regulate the quorum sensing response [16, 17]. Environmental factors are important to understand the formation of biofilms, which act to protect bacteria and enhance the efficacy of their virulence. To identify effective quorum sensing inhibition therapies, LuxIR-type models have been used to examine the robustness of the quorum sensing response as a function of the depth of the biofilm and in terms of other parameters that describe the stability of the quorum sensing system [1, 2]. These and other studies have contributed to the understanding and development of different quorum sensing inhibition strategies to develop novel antivirulence therapies [1, 2, 3, 30, 31].

More recently, however, complex *in silico* simulations of LuxIR-type circuits are proving useful to test the three fundamental assumptions of quorum sensing systems discussed in Section 2.1. Specifically, researchers have shown that LuxIR-type circuits act to synchronize heterogenous populations [43, 92, 98] and facilitate a robust on-off switch [39, 38, 43, 97]. Lastly, bioengineering and biomedical applications of this research are also being explored such as engineering synthetic cells with different cell-population density sensors [40].

2.5 Quorum Sensing in Hybrid Circuits

Unlike LuxIR-type quorum sensing circuits, there is no detailed model of the hybrid quorum sensing system in *V. harveyi* and *V. cholerae*. Therefore, in this section, a novel mathematical model of the *V. harveyi* and *V. cholerae* quorum sensing system is formulated based on the understanding of these systems given in Section 1.1. In what follows, a set of differential equations that model the reaction kinetics of the sRNA circuit is derived. Although the focus of this work is on the sRNA circuit in *V. harveyi* and *V. cholerae*, a simple model of the *V. harveyi* phosphorelay cascade is needed to incorporate more experimental data into the parameterization of the model of the *V. harveyi* sRNA circuit.

2.5.1 A Model of the Phosphorelay Cascade

Swem et al. parameterized a model of the *V. harveyi* autoinducer receptors and found that the difference in free energy between the kinase and phosphatase states is

$$\frac{\Delta G}{k_B T} = -2.3 + \ln\left(\frac{1+AI}{1+10^{-6}AI}\right), \tag{2.19}$$

where AI is the concentration (nM) of autoinducer [96]. Assuming there is only one phosphorelay cascade, the input for the sRNA circuit is the ratio of LuxO-P, O_P, to LuxO, O [62]. At steady state, $O_P = \Gamma O$, where the equilibrium constant, Γ, is of the form $\Gamma = \exp\left(-\frac{\Delta G}{k_B T}\right)$. Therefore, the simple model of the phosphorelay cascade is

$$\Gamma \quad = \quad \exp\left(2.3 - \ln\left(\frac{1+AI}{1+10^{-6}AI}\right)\right). \tag{2.20}$$

If the autoinducer concentration is known, then (2.20) can be used to relate the concentration of autoinducer to Γ, otherwise Γ is a parameter representative of the cell-population density. Note that LCD corresponds to large Γ, while HCD corresponds to small Γ.

2.5.2 A Novel Model of the Small RNA Circuit

In this section, a novel model of the sRNA circuit is derived following from the overview provided in Section 1.1. Transcription of *luxO* is inhibited by LuxO, O, and LuxO-P, O_P and the rates at which *luxO* mRNA, o, is translated and decays proportional to its concentration. LuxO decays at a rate proportional to its concentration as well.

$$\begin{array}{ccc} \xrightarrow{\kappa_o(O,O_P)} & o & \xrightarrow{\delta_o} \\ \xrightarrow{\kappa_O} & O & \xrightarrow{\delta_O} \end{array} \tag{2.21}$$

We assume that there is a basal rate of expression of *luxO* that is inhibited equally by both LuxO and LuxO-P [103]. Therefore, the transcription rate of *luxO* is

$$\kappa_o(O, O_P) \quad = \quad \frac{V_o}{1 + K_O(O + O_P)}. \tag{2.22}$$

The reactions governing the expression of *luxR*/*hapR* are identical in form to those for *luxO*:

$$\begin{array}{ccc} \xrightarrow{\kappa_r(R)} & r & \xrightarrow{\delta_r}, \\ \xrightarrow{\kappa_R} & R & \xrightarrow{\delta_R}. \end{array} \tag{2.23}$$

The main difference is that the *luxR* transcription rate is partially inhibited by a LuxR dimer [61, 15]. This is also assumed to be the case for HapR given that they are from the same family of transcriptional regulators [81]. The *luxR*/*hapR* transcription rate is, therefore,

$$\kappa_r(R) \quad = \quad V_{r0} + \frac{V_r}{1 + (K_r R)^2}. \tag{2.24}$$

The reactions governing the expression of the n'th species of *qrr* are summarized in Figure 2.3. LuxO-P activates *qrr* expression and a LuxR/HapR dimer enhances this expression. To model this process, four different states for the *qrr* promoter are introduced to represent the probability that the promoter is unbound, P_n, bound by a LuxR/HapR dimer, P_{R_n}, bound by LuxO-P, P_{O_n}, or bound by LuxO-P and a LuxR/HapR dimer, P_{RO_n} [77]. The rates of *qrr* transcription for the latter two states are also assumed to be different.

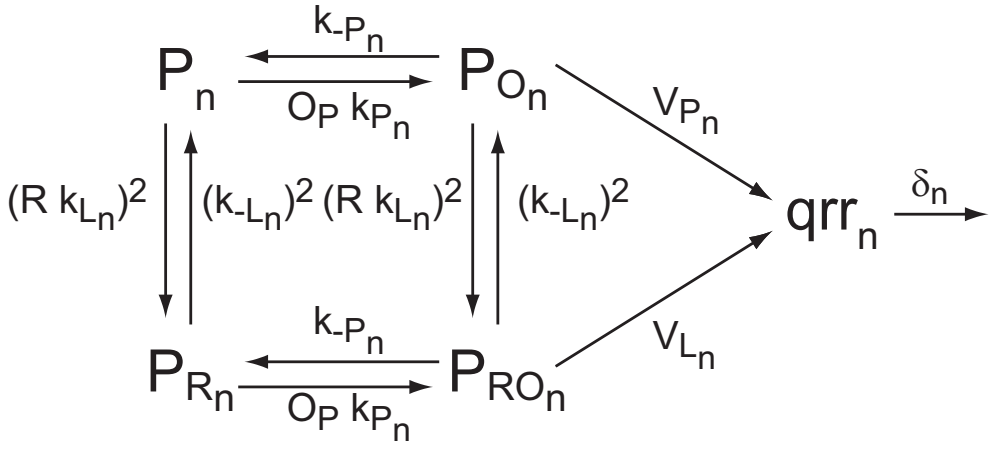

Figure 2.3. Qrr promoter model. The four states represent the probability that the promoter is bound by LuxR and/or LuxO-P.

The corresponding equations governing the states of the *qrr* promoter are:

$$1 = P_{R_n} + P_{O_n} + P_{OR_n} + P_n, \tag{2.25}$$

$$\frac{\mathrm{d}P_{R_n}}{\mathrm{d}t} = (k_{L_n}R)^2 P_n + k_{-P_n} P_{RO_n} - (k_{-L_n}^2 + O_P k_{P_n}) P_{R_n}, \tag{2.26}$$

$$\frac{\mathrm{d}P_{O_n}}{\mathrm{d}t} = k_{P_n} O_P P_n + (k_{-L_n})^2 P_{RO_n} - ((Rk_{L_n})^2 + k_{-P_n}) P_{O_n}, \tag{2.27}$$

$$\frac{\mathrm{d}P_{RO_n}}{\mathrm{d}t} = k_{P_n} O_P P_{R_n} + (Rk_{L_n})^2 P_{O_n} - ((k_{-L_n})^2 + k_{-P_n}) P_{O_n}, \tag{2.28}$$

$$\frac{\mathrm{d}q_n}{\mathrm{d}t} = V_{P_n} P_O + V_{L_n} P_{RO_n} - \delta_n q_n. \tag{2.29}$$

Solving for the steady state probabilities P_{RO_n} and P_{O_n} and rewriting (2.29) gives

$$\frac{\mathrm{d}q_n}{\mathrm{d}t} = \frac{K_{P_n} O_P}{1 + K_{P_n} O_P} \frac{V_{P_n} + V_{L_n}(K_{L_n}R)^2}{1 + (K_{L_n}R)^2} - \delta_n q_n. \tag{2.30}$$

The final reactions for the sRNA model relate to the formation of Hfq-Qrr, H_n, and the repression of LuxO and LuxR/HapR from Hfq-Qrr, as summarized in (2.31) – (2.33). The total concentration of Hfq available for quorum sensing is assumed constant, H_0, because Hfq is pleiotropic and abundant in cells [36, 12]. Hfq acts to stabilize and help Qrr bind target mRNA. Hfq releases the sRNA-mRNA pair, which is then degraded while Hfq remains intact [57]. The net result of the reaction is the loss of one sRNA for every mRNA.

$$H_0 + q_n \xrightarrow{\beta_n} H_n \tag{2.31}$$

$$H_n + o \xrightarrow{\nu_n} H_0 \tag{2.32}$$

$$H_n + r \xrightarrow{\mu_n} H_0 \tag{2.33}$$

In summary, the complete set of equations governing the sRNA circuit is,

$$\Gamma = \exp\left(2.3 - \ln\left(\frac{1+AI}{1+10^{-6}AI}\right)\right), \tag{2.34}$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \kappa_R r - \delta_R R, \tag{2.35}$$

$$\frac{\mathrm{d}O}{\mathrm{d}t} = \kappa_O o - \delta_O O, \tag{2.36}$$

$$\frac{\mathrm{d}r}{\mathrm{d}t} = V_{r0} + \frac{V_r}{1+(K_R R)^2} - \sum_{i=1}^{4} \mu_i H_i r - \delta_r r, \tag{2.37}$$

$$\frac{\mathrm{d}o}{\mathrm{d}t} = \frac{V_o}{1+K_O(1+\Gamma)O} - \sum_{i=1}^{4} \nu_i H_i o - \delta_o o, \tag{2.38}$$

$$\frac{\mathrm{d}q_n}{\mathrm{d}t} = \frac{K_{P_n}\Gamma O}{1+K_{P_n}\Gamma O}\frac{V_{P_n}+V_{L_n}(K_{L_n}R)^2}{1+(K_{L_n}R)^2} - \beta_n\left(H_0 - \sum_{i=1}^{4} H_i\right) q_n - \delta_n q_n, \tag{2.39}$$

$$\frac{\mathrm{d}H_n}{\mathrm{d}t} = \beta_n\left(H_0 - \sum_{i=1}^{4} H_i\right) q_n - \mu_n H_n r - \nu_n H_n o, \tag{2.40}$$

where $n = 1...4$ corresponds to the index of nth species of sRNA. There are only four rather than five Qrr since *qrr5* is not quorum regulated in *V. harveyi*.

The nondimensionalization of the equations is used to simplify their parameterization. First, the following are characteristic concentrations:

$$r_M = \frac{V_{r0}+V_r}{\delta_r}, \qquad o_0 = \frac{V_o}{\delta_o}, \qquad O_0 = \frac{\kappa_O}{\delta_O}o_0, \qquad R_0 = \frac{\kappa_R}{\delta_R}r_M, \qquad Q_n = \frac{V_{P_n}}{\delta_n}. \tag{2.41}$$

The variables in the model are rescaled using the characteristic concentrations

$$r = r_M\widehat{r}, \qquad R = R_0\widehat{R}, \qquad H_n = H_0\widehat{H}_n, \tag{2.42}$$

$$o = o_0\widehat{o}, \qquad O = O_0\widehat{O}, \qquad q_n = Q_n\widehat{q}_n. \tag{2.43}$$

Next, define the dimensionless parameters

$$\widehat{K}_{P_n} = K_{P_n}O_0, \qquad \widehat{K}_{L_n} = K_{L_n}R_0, \qquad \widehat{K}_O = K_O O_0, \qquad \widehat{K}_R = K_R R_0, \tag{2.44}$$

$$E_{q_n} = \frac{H_0\beta_n}{\delta_n}, \qquad E_{r_n} = \frac{H_0\mu_n}{\delta_r}, \qquad E_{o_n} = \frac{H_0\nu_n}{\delta_o}, \tag{2.45}$$

$$V_{q_n} = \frac{V_{L_n}}{V_{P_n}}, \qquad V_{r_n} = \frac{V_{r0}+V_r}{V_{P_n}}, \qquad V_{or} = \frac{V_o}{V_{r0}+V_r}, \qquad r_0 = \frac{V_{r0}}{V_{r0}+V_r}. \tag{2.46}$$

At steady state, the model simplifies to

$$\Gamma = \exp\left(2.3 - \ln\left(\frac{1+AI}{1+10^{-6}AI}\right)\right), \tag{2.47}$$

$$0 = r_0 + \frac{1-r_0}{1+(\widehat{K}_R r)^2} - \left(\sum_{n=1}^{4} E_{r_n}H_n + 1\right) r, \tag{2.48}$$

$$0 = \frac{1}{1+\widehat{K}_O(1+\Gamma)o} - \left(\sum_{n=1}^{4} E_{o_n} H_n + 1\right) o, \tag{2.49}$$

$$0 = \frac{\widehat{K}_{P_n}\Gamma o}{1+\widehat{K}_{P_n}\Gamma o}\frac{1+V_{q_n}(\widehat{K}_{L_n} r)^2}{1+(\widehat{K}_{L_n} r)^2} - \left(E_{q_n}\left(1-\sum_{m=1}^{4} H_m\right)+1\right) q_n, \tag{2.50}$$

$$0 = E_{q_n}\left(1-\sum_{m=1}^{4} H_m\right) q_n - V_{r_n}\left(E_{r_n} r + V_{or} E_{o_n} o\right) H_n. \tag{2.51}$$

$\widehat{K}_R$ and $\widehat{K}_O$ represent the LuxR/HapR and LuxO autoregulation, respectively. V_{q_n} and $\widehat{K}_{L_n}$ represent the LuxR/HapR-Qrr feedback, and E_{o_n} and V_{or} represent the LuxO-Qrr feedback. For simplicity, the "$\widehat{\ }$" notation on o, r, q_n, and H_n is dropped. A summary of the interpretation of the parameters is given in Table 2.1.

2.5.3 The Contribution of Hybrid Quorum Sensing Mathematical Models

Mathematical models of the hybrid *V. harveyi* and *V. cholerae* quorum sensing systems have been formulated to answer questions associated with either the phosphorelay cascade or the sRNA circuit. The first model of the phosphorelay cascade in *V. harveyi* showed that

Table 2.1. Interpretation of the nondimensional parameters.

Parameter	Interpretation
Γ	Steady state ratio of LuxO-P to LuxO (a measure of the local cell-population density).
$\widehat{K}_R$	Affinity of LuxR/HapR to its promoter (LuxR/HapR autoregulation).
$\widehat{K}_O$	Affinity of LuxO to its promoter (LuxO autoregulation).
$\widehat{K}_{L_n}$	Affinity of LuxR/HapR to the *qrr* promoter bound with LuxO-P (part of the LuxR/HapR-Qrr feedback).
V_{q_n}	Rate of *qrr* expression from LuxR/HapR relative to the rate of *qrr* expression from LuxO-P only (part of the LuxR/HapR-Qrr feedback).
E_{o_n}	Rate at which *qrr* binds *luxO* mRNA relative to the rate at which *luxO* mRNA decays (the LuxO-Qrr feedback).
V_{or}	Rate of *luxO* expression relative to the rate of *luxR/hapR* expression.
V_{r_n}	Rate of *luxR/hapR* expression relative to the rate of *qrr* expression from LuxO-P.
$\widehat{K}_{P_n}$	Affinity of LuxO-P to the *qrr* promoter.
E_{r_n}	Rate at which *qrr* binds *luxR/hapR* mRNA relative to the rate at which *luxR/hapR* mRNA decays.
E_{q_n}	Affinity of *qrr* sRNA to Hfq relative to its decay.
r_0	Basal rate of *luxR/hapR* expression.

the autoinducer receptors can be described by a two state model [96]. Subsequent studies have shown that the autoinducer receptors phosphorylate LuxO in a graded, proportionate, and additive manner [7, 65, 62]. Therefore, the *V. harveyi* and *V. cholerae* quorum sensing system regulates genes downstream of the quorum sensing system based on the total level of LuxO-P and that each autoinducer is responsible for approximately 1/3 of LuxO phosphorylation. Furthermore, these empirical studies also show that the phosphorelay cascade does not facilitate an ultrasensitive or bistable response during the LCD to HCD transition, which is different than LuxIR-type circuits.

The discovery of the sRNA circuit led some to argue that the sRNA circuit can facilitate a robust, ultrasensitive response that acts to synchronize a population, in support of the fundamental assumptions of quorum sensing systems. Hence, many of the sRNA models have been developed to test these assumptions, as summarized in Section 2.2. Only two studies have addressed the sRNA circuit in *V. harveyi* and/or *V. cholerae*. Jian-Wei showed that *V. harveyi* can exhibit periodic oscillations in bioluminescence when a time delay is incorporated into the model [49]. Although interesting on its own, such behavior has not been observed in empirical studies and a biological basis for such a delay mechanism is lacking. The second sRNA model is summarized in Section 4.1, for it relates closely to the work presented in Chapter 4. In the interest of simplicity, we note that the formulation of all of these models has excluded the Qrr feedback and Hfq. Therefore, the model in Section 2.5.2 is the only such model that can test the dosage compensation hypothesis of [94] and, as will be evident in Chapter 4, can directly show the importance of Hfq.

CHAPTER 3

A MATHEMATICAL MODEL AND QUANTITATIVE COMPARISON OF THE SMALL RNA CIRCUIT IN THE *VIBRIO HARVEYI* AND *VIBRIO CHOLERAE* QUORUM SENSING SYSTEMS

The similarities between the *V. harveyi* and *V. cholerae* quorum sensing systems make it difficult to identify the mechanisms underlying kinetic differences between the species. In this chapter, the parameters for model of the *V. harveyi* and *V. cholerae* sRNA circuit introduced in Section 2.5 are estimated by fitting the model to a variety of empirical data from both species. All of the parameters can be distinguished and the parameter estimation is robust to errors in the data. This results in a model that can, therefore, be used for *in silico* design, testing, and analysis of experiments. An example of such an experiment suggests that *V. cholerae* Qrr are more abundant and more sensitive to changes in LuxO than *V. harveyi* Qrr. This could explain why expression of HapR is more robust than LuxR to changes in Qrr. Although a few weak search directions are identified in the parameter estimation of *V. harveyi* and *V. cholerae*, one suggested utility of the model is that it can be used to identify a series of experiments that contain new information about the parameters and, hence, complete the model. The results in this chapter are used in Chapter 4 to identify the mechanisms underlying the additive and redundant Qrr phenotypes in *V. harveyi* and *V. cholerae*.

3.1 Introduction

In this work, the mathematical model of the *V. harveyi* and *V. cholerae* sRNA circuit is used to identify and explain the mechanisms underlying some kinetic differences between

V. harveyi and *V. cholerae*. Empirical data are used to solve a constrained, nonlinear least squares problem to estimate the 35 and 33 parameters, respectively, in the *V. harveyi* and *V. cholerae* sRNA model. The parameter estimation is implemented in Matlab using the nonlinear least-squares solver `lsqnonlin` and exact Jacobian of the forward map, which is a model of the measurements based on the sRNA model. The behavior of the model agrees quantitatively with the all of the empirical data available and is, therefore, representative of the *V. harveyi* and *V. cholerae* sRNA circuits.

A series of simple experiments are then proposed that highlight kinetic differences between the species. It was shown that Qrr are more abundant in *V. cholerae* than in *V. harveyi* and that *V. harveyi* and *V. cholerae* Qrr are sensitive to changes in LuxR and LuxO, respectively. We argue and demonstrate that this explains why dosage compensation is stronger in *V. cholerae* than in *V. harveyi*. These results refine the hypothesis of Svenningsen et al. who suggested that the differences in LuxR/HapR repression is a consequence of stronger dosage compensation in *V. cholerae* than in *V. harveyi* [94]. Lastly, the model suggests that the saturation of Hfq, a protein chaperon that stabilizes Qrr, with Qrr is essential for the robust repression of target mRNA.

3.2 Results and Discussion

In this section, the empirical data from *V. harveyi* and *V. cholerae* are used to parameterize the quorum sensing model introduced in Chapter 2. Furthermore, the model with a parameterization for each species agrees well with the data, showing that the model is representative of quorum sensing in *V. harveyi* and *V. cholerae*. Lastly, the model is used to predict novel behavior in *V. harveyi* and *V. cholerae*. To parameterize the model, the following constrained, nonlinear least squares problem is solved

$$\min_{\boldsymbol{p}\geq\boldsymbol{a}} ||\boldsymbol{F}(\boldsymbol{p}) - \boldsymbol{d}||^2. \tag{3.1}$$

Here, $\boldsymbol{d}$ is a vector containing the measurements from the experiments and $\boldsymbol{p}$ is a vector representing the wild-type parameterization. The constraint $\boldsymbol{p} \geq \boldsymbol{a}$ is necessary to ensure that $V_{r_n} \geq 1$ for all n (i.e., so that LuxR/HapR only enhances Qrr expression) and that all of the remaining parameters are non-negative. $F_i(\boldsymbol{p})$ is a model of the experiment corresponding to the i'th measurement. All of the models of the experiments are stored together in the vector $\boldsymbol{F}(\boldsymbol{p})$. Therefore, $F_i(\boldsymbol{p}) - d_i$ is the error associated with modeling the i'th experiment, while $||\boldsymbol{F}(\boldsymbol{p}) - \boldsymbol{d}||^2$ represents the total error between the model and all of the experiments for the given wild-type parameterization. A detailed discussion of the data and how they were modeled is provided in the sections that follow. The problem was solved

using Matlab's `lsqnonlin` function. To improve the accuracy and rate of convergence, the Jacobian of $\boldsymbol{F}(\boldsymbol{p})$ was calculated exactly by differentiating (2.48)–(2.51) and using these derivatives to compute $\nabla F_i(\boldsymbol{p})$.

To find the global minimum, the problem was solved using several different initial guesses that spanned a feasible set containing the solution. Each initial guess is a vector of uniformly distributed random numbers generated over the feasible set of wild-type parameters. To find a reasonable feasible set for all of the parameters, a large feasible set is initialized then manually refined until it was as small as possible but large enough so that it contained the solution to each randomly generated parameter vector. The nonlinear least-squares solver was terminated either when the residual was below a certain threshold (i.e., $||\boldsymbol{F}(\boldsymbol{p}) - \boldsymbol{d}||^2 \leq 10^{-4}$) or after some finite number of iterations. The parameterization for each species and corresponding final feasible set is summarized in Table 3.1.

The next two sections describe the experiments and how they were modeled. Although the details of $F_i(\boldsymbol{p})$ are different, they all have the following general structure. Each mutant strain in the experiment was parameterized by modifying the wild-type parameterization accordingly. For example, setting $E_{o_n} = V_{or} = 0$ in the wild-type parameterization represents a strain without the LuxO-Qrr feedback. Next, the steady state solution of each strain was computed by solving (2.48)–(2.51). The exact Jacobian of $\boldsymbol{F}(\boldsymbol{p})$ was also used to decrease running time and improve the accuracy of the nonlinear solver. Lastly, steady state quantities in the model that corresponded to the quantities measured in the experiments were measured (such as the ratio of the steady state concentration of $luxR/hapR$ in a wild-type strain relative to a mutant strain).

3.2.1 *V. harveyi* Parameterization

In this section, the *V. harveyi* data that were used to parameterize the model is described. The first two experiments below are used to parameterize r_0, K_R, and K_{L_n} $(n = 1, 2, 3, 4)$ because those data are uniquely determined by those parameters. K_R and K_{L_n} are related to their dimensionless counterparts by the characteristic concentration of LuxR, R_0. The rest of the parameters were fit simultaneously to the remaining data using the formulation described by 3.1 by treating R_0 as a parameter rather than $\widehat{K}_{L_n}$ and $\widehat{K}_R$. The full *V. harveyi* parameterization is shown in Table 3.1.

3.2.1.1 LuxR Autoregulation

Chatterjee et al. identified the regions of the *luxR* promoter involved in the autoregulation of LuxR and used mobility-shift assays to measure the proportion of *luxR* promoters

Table 3.1. Parameters and their corresponding feasible set for *V. harveyi* and *V. cholerae*. All parameters are dimensionless except for R_0 as indicated.

Parameter	*V. harveyi*		*V. cholerae*	
	Value	Feasible Set	Value	Feasible Set
r_0	$3.67{\cdot}10^{-1}$	–	$2.96{\cdot}10^{-1}$	[0, 0.6]
$\widehat{K}_R$	$1.38{\cdot}10^{1}$	–	$9.03{\cdot}10^{-1}$	[0, 1]
$\widehat{K}_O$	1.80	[1, 7]	$1.55{\cdot}10^{1}$	[5, 35]
Γ_{LCD}	$1.82{\cdot}10^{-2}$	[0, 1]	$2.15{\cdot}10^{-1}$	[0.15, 0.35]
Γ_{HCD}	$1.94{\cdot}10^{-1}$	[0, 0.07]	–	–
$\widehat{K}_{P_1}$	$3.62{\cdot}10^{-1}$	[0, 3]	2.65	[2, 3]
$\widehat{K}_{P_2}$	3.60	[1, 11]	$3.94{\cdot}10^{1}$	[26, 42]
$\widehat{K}_{P_3}$	7.30	[2, 20]	$9.36{\cdot}10^{-1}$	[0.7, 1]
$\widehat{K}_{P_4}$	1.13	[0, 7]	8.89	[7, 10]
$\widehat{K}_{L_1}$	0	–	$1.84{\cdot}10^{2}$	[120, 260]
$\widehat{K}_{L_2}$	$2.21{\cdot}10^{1}$	–	$3.04{\cdot}10^{1}$	[25, 55]
$\widehat{K}_{L_3}$	$1.38{\cdot}10^{1}$	–	$1.08{\cdot}10^{1}$	[8.5, 12]
$\widehat{K}_{L_4}$	$2.95{\cdot}10^{1}$	–	$4.77{\cdot}10^{1}$	[33, 50]
E_{r_1}	$5.05{\cdot}10^{5}$	$[1{\cdot}10^{5}, 9{\cdot}10^{5}]$	$8.36{\cdot}10^{1}$	[35, 95]
E_{r_2}	$1.83{\cdot}10^{3}$	[250, 2000]	$2.87{\cdot}10^{2}$	[55, 320]
E_{r_3}	$2.31{\cdot}10^{1}$	[3 70]	$2.98{\cdot}10^{2}$	[150, 500]
E_{r_4}	$1.08{\cdot}10^{3}$	$[250, 2.2{\cdot}10^{3}]$	$4.76{\cdot}10^{1}$	[40, 700]
E_{o_1}	$1.35{\cdot}10^{-2}$	[0, 5]	$1.07{\cdot}10^{1}$	[2.5, 12]
E_{o_2}	$1.20{\cdot}10^{1}$	[0, 25]	$6.99{\cdot}10^{1}$	[14, 80]
E_{o_3}	$2.33{\cdot}10^{2}$	[75, 300]	$1.21{\cdot}10^{2}$	[100, 400]
E_{o_4}	$8.18{\cdot}10^{-1}$	[0, 4]	$2.95{\cdot}10^{-1}$	[0, 0.3]
E_{q_1}	$4.65{\cdot}10^{-3}$	[0, 0.2]	$2.98{\cdot}10^{-1}$	[0, 3]
E_{q_2}	3.31	[0, 35]	$2.31{\cdot}10^{-3}$	[0, 0.8]
E_{q_3}	9.57	[5, 40]	$1.58{\cdot}10^{2}$	[0, 180]
E_{q_4}	$1.74{\cdot}10^{-1}$	[0, 1.25]	$4.33{\cdot}10^{1}$	[0, 75]
V_{q_1}	0	–	4.99	[4, 6]
V_{q_2}	1.90	[1.5, 2.2]	1.91	[1.25, 2.5]
V_{q_3}	2.28	[2.1, 2.5]	$1.41{\cdot}10^{1}$	[12, 16]
V_{q_4}	2.56	[2.4, 2.8]	3.65	[3, 4.5]
V_{r_1}	1.44	[1, 5]	$7.81{\cdot}10^{-2}$	[0, 0.45]
V_{r_2}	5.80	[0, 7]	$8.49{\cdot}10^{-4}$	[0.15, 0.45]
V_{r_3}	$5.83{\cdot}10^{-1}$	[0 2]	$2.51{\cdot}10^{-2}$	[0, 0.05]
V_{r_4}	4.63	[2, 5]	$5.57{\cdot}10^{-1}$	[0.2, 0.8]
V_{or}	$1.28{\cdot}10^{1}$	[2, 22]	2.42	[1, 5]
R_0 (nM)	$5.16{\cdot}10^{2}$	[200, 600]	–	–

bound at a given concentration of LuxR [15]. These data were used to parameterize the *luxR* promoter model, $r_0 + (1 - r_0)/(1 + (K_R R)^2)$. The data (dots) and the results from the parameterization (solid curve) are shown in Figure 3.1. The results suggest that $K_R \approx 0.0250(\mu g)^{-1}$ and that $r_0 \approx 0.38$.

3.2.1.2 LuxR Affinity to the *qrr* Promoters

Tu et al. used mobility-shift assays to show that LuxR enhances *qrr* expression by binding directly to each *qrr* promoter [104]. We set $K_{L_1} = 0\text{nM}^{-1}$, $K_{L_2} = (25\text{ nM})^{-1}$, $K_{L_3} = (40\text{nM})^{-1}$, and $K_{L_4} = (19\text{nM})^{-1}$ based on visual inspection of the data in Figure 2 of their work [104].

3.2.1.3 LuxR-Qrr Feedback at LCD and HCD

Tu et al. showed that LuxR enhances Qrr expression in *V. harveyi* when it binds the *qrr* promoter [104]. They created a $\Delta luxR$ and a $qrr2,3,4^{luxR-bs}$ strain, which has a scrambled LuxR binding site in each *qrr* promoter to limit/prevent LuxR binding. Using quantitative real-time PCR analysis, they measured the level of *qrr* at low and at high cell density in a wild-type strain and each mutant strain. They present their results by normalizing the concentration of *qrr* by their corresponding wild-type concentration at LCD. The data, shown in Figure 3.2 (left), show that LuxR enhances *qrr2-4* expression and that there is

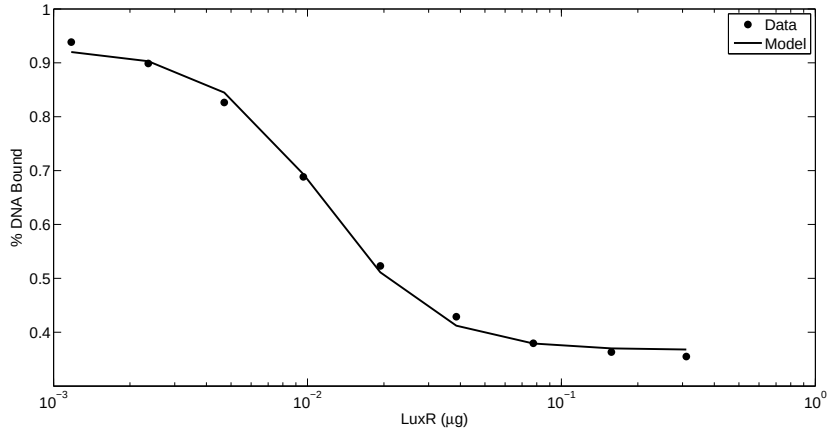

Figure 3.1. A comparison between the data (dotted points) and the behavior of the $luxR$ promoter model (solid curve) of the proportion of LuxR promoters bound by LuxR as the concentration of LuxR varies. The data were generated using mobility-shift assays and taken from Figure 6A of [15] and shows that LuxR binds its own promoter. These data were used to parameterize the $luxR$ promoter model, $r_0 + (1 - r_0)/(1 + (K_R R)^2)$, and found that $K_R = 0.0250(\mu g)^{-1}$ and $r_0 = 0.38$.

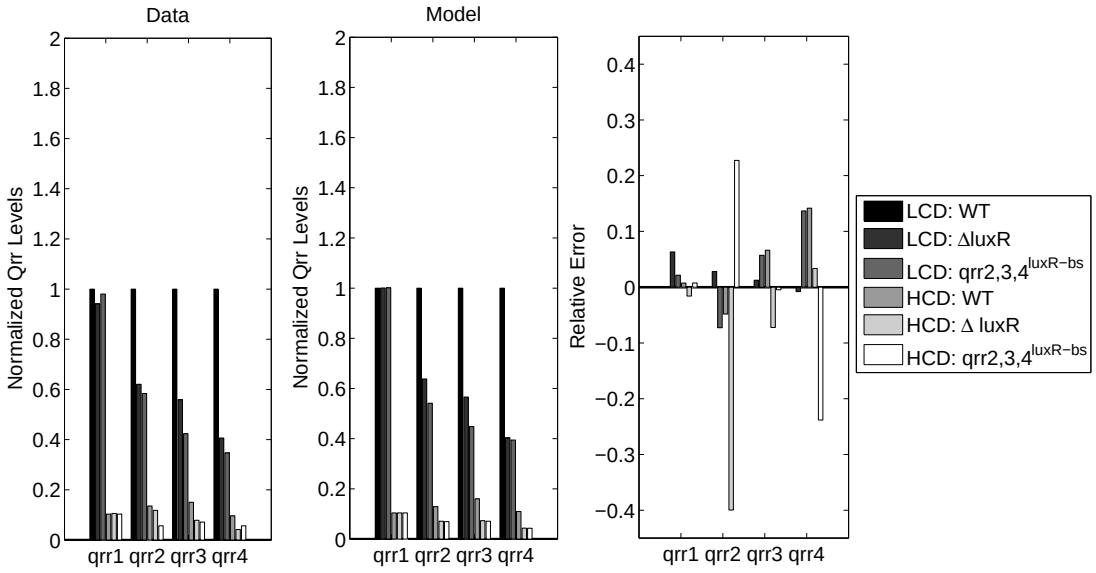

Figure 3.2. Comparison between the data (left) and the results (center) of the fold change in the concentration of each Qrr in the absence of the LuxR-Qrr feedback at LCD and HCD. Three strains were used in the experiments: wild-type, $\Delta luxR$, and $qrr2,3,4^{luxR-bs}$ (one with a scrambled LuxR binding site in each Qrr promoter). The concentration of Qrr was measured in each strain at LCD and HCD and normalized by the concentration in the wild-type strain at LCD. The relative error of the results is shown on the right.

little difference in *qrr* concentration between the mutant strains.

To model this experiment, the wild-type parameterization is modified to model the two mutant strains: $\Delta luxR$ ($E_{r_n} = \widehat{K}_{L_n} = V_{q_n} = 0$), and $qrr2,3,4^{luxR-bs}$ ($\widehat{K}_{L_n} = V_{q_n} = 0$). Two different values of Γ are also estimated that correspond to the different ratios of LuxO:LuxO-P at low and at high cell density (i.e., $\Gamma_{LCD} > \Gamma_{HCD}$). For each strain, the steady state concentration of each *qrr* at Γ_{LCD} and at Γ_{HCD} is computed. Lastly, each *qrr* concentration is normalized by its corresponding concentration in the wild-type strain at LCD. The final results (middle) and corresponding error (right) are shown in Figure 3.2. The model agrees well with the data at both low and high cell density, although there is less agreement at HCD.

3.2.1.4 Role of LuxO Regulation in *V. harveyi*

Tu et al. showed that LuxO regulation affects the onset of the LCD to HCD transition and the dynamic range of expression of quorum sensing target gene expression [103]. They introduced a LuxR-mCherry protein fusion into the *V. harveyi* chromosome at the native *luxR* locus in four different strains: wild-type, −LuxO Auto, −LuxO-Qrr feedback, −LuxO

regulation. The latter three strains lack LuxO autoregulation, LuxO-Qrr feedback, or both, respectively. They used single cell fluorescence microscopy to measure LuxR-mCherry in individual cells over a range of autoinducer concentrations for each strain as a means to infer *luxR* expression. Their results, in Figure 3.3 (left), show that the onset of the LCD to HCD transition is shifted to larger autoinducer concentrations as LuxO regulation is removed. The data also show that there is little difference in LuxR expression between the −LuxO Auto and −LuxO-Qrr feedback strains.

To model this experiment, parameterizations of each strain were created. Starting with the wild-type parameterization, we set $\widehat{K}_O = 0$ for the −LuxO Auto strain, $E_{o_n} = V_{or} = 0$ for the −LuxO-Qrr feedback strain, and $\widehat{K}_O = E_{o_n} = V_{or} = 0$ for the −LuxO regulation strain. Since fluorescence is expressed as a function of autoinducer concentration, (2.20) is used to relate Γ to the concentration of autoinducer in the data. Then, the steady state concentration of LuxR in each strain at every autoinducer concentration is computed. The results (middle) and corresponding error (right) are shown in Figure 3.3. The results reproduce the shift in the LuxR dose response curve for the various mutant strains. Note

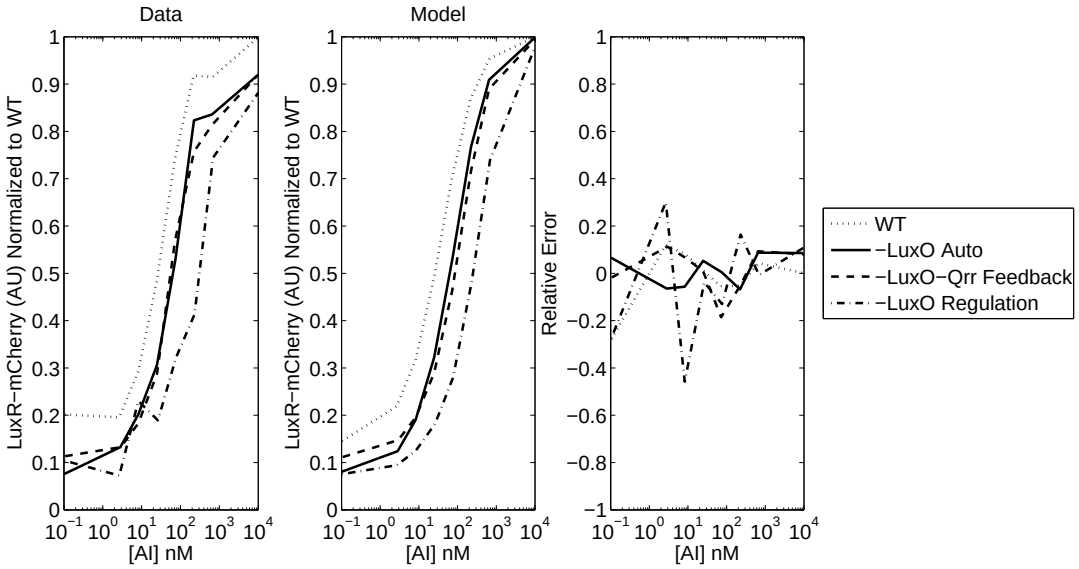

Figure 3.3. A comparison between the data [103] (left) and the results (middle) showing the florescence of the LuxR-mCherry construct over increasing autoinducer concentrations in wild-type, −LuxO-Auto, −LuxO-Qrr feedback, and −LuxO regulation strains. The data and the results are normalized to the LuxR-mCherry florescence in the wild-type strain at an autoinducer concentration of 10^4 nM. The error associated with the results (right) is largest when autoinducer concentrations are small, which also corresponds to when there is more uncertainty in the data.

that the error is largest at LCD for which there is more uncertainty in the data as well.

Tu et al. repeated the experiment using strains with *qrr4* only. Their results in Figure 3.4 (left) show a shift in the onset of the LCD to HCD transition similar to their previous results. Additionally, there is a 3 rather than a 5-fold change in fluorescence from LCD to HCD. To model this experiment, the previous experiment was repeated but with $\widehat{K}_{P_1} = \widehat{K}_{P_2} = \widehat{K}_{P_3} = 0$ in all of the strains. The results, Figure 3.4 (middle), reflect a similar shift in the onset as the data; however, the results also show more repression of LuxR at LCD than what is reflected in the data.

3.2.2 *V. cholerae* Parameterization

The *V. cholerae* parameters for the model were estimated using all of the *V. cholerae* data by solving the problem described by (3.1). The *V. cholerae* experiments were all performed at the same optical density corresponding to LCD so there is only one value of Γ in the *V. cholerae* parameterization. In what follows, the four *V. cholerae* experiments, how each was modeled, and the results are discussed. Table 3.1 lists the *V. cholerae* parameters.

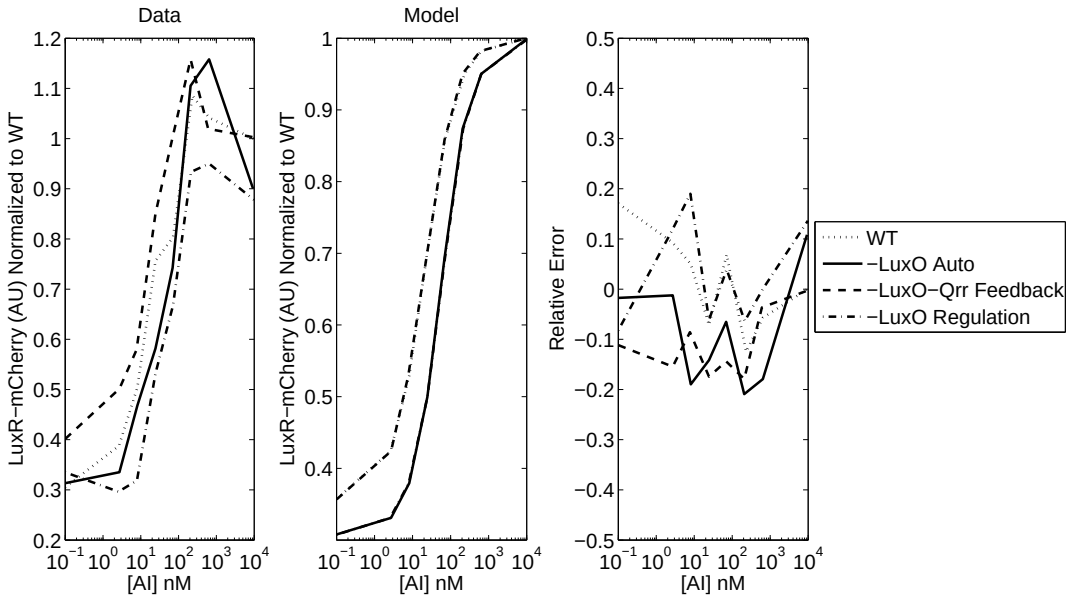

Figure 3.4. A comparison between the data [103] (left) and the results (middle) showing the florescence of the LuxR-mCherry construct over different autoinducer concentrations in wild-type, −LuxO-Auto, −LuxO-Qrr feedback, and −LuxO regulation strains containing *qrr*4 only. The data and the results are normalized to the LuxR-mCherry florescence in the wild-type strain at an autoinducer concentration of 10^4 nM. The error (right) shows that the repression of LuxR at low autoinducer concentrations in the model is much larger than that observed in the data.

3.2.2.1 HapR Repression

Svenningsen et al. showed that one *qrr* is sufficient to repress *hapR* to near wild-type levels [94]. They created four mutant strains that had only one type of *qrr* and a mutant strain without any *qrr*. Using real-time PCR analysis, they measured *hapR* concentration in each strain and normalized the *hapR* concentration by its concentration in the wild-type strain. Their results, Figure 3.5 (left), show that all Qrr significantly repress *hapR* similar to wild-type levels especially *qrr*4.

To model this experiment, the wild-type parameterization was used with $\widehat{K}_{P_n} = 0$ for each of the n *qrr* knocked out in the mutant strains, i.e., for the $+qrr2$ strain, we let $\widehat{K}_{P_n} = 0$ for $n = 1, 3, 4$. To parameterize the Δqrr strain, $\widehat{K}_{P_n} = 0$ for all n. The steady state concentration of *hapR* in each strain was found then normalized each by the *hapR* concentration in the wild-type strain. A comparison between the data (left), model (center), and the relative error (right) is shown in Figure 3.5. The results show that the model agrees well with the data.

3.2.2.2 Dosage Compensation

Svenningsen et al. showed that *qrr* expression increases in the absence of one or more *qrr* in *V. cholerae* and called this phenomenon dosage compensation [94]. Using real-time PCR

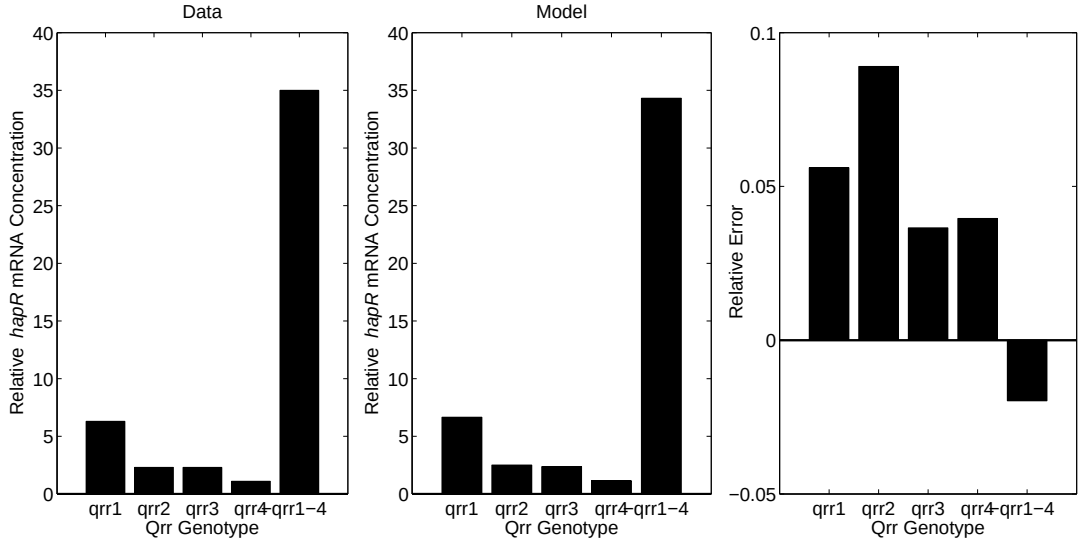

Figure 3.5. Comparison between the data [94] (left) and the results (middle) showing the fold change in LCD *hapR* mRNA concentration in a strain with at most one Qrr relative to *hapR* mRNA concentration a wild-type strain. The error (right) shows that the model is in good agreement with the data.

analysis, Svenningsen et al. measured the concentrations of *hapR* and *qrr* in a wild-type, $\Delta qrr3$, $\Delta qrr2,3$, and in a $\Delta qrr1,2,3$ strain at LCD. These data were then normalized to their corresponding wild-type levels. Their results show that, as each *qrr* is removed, the expression of the remaining *qrr* increases, while *hapR* remains relatively constant [94].

This experiment is modeled as follows. For the mutant strains, the wild-type parameterization is modified by setting $\widehat{K}_{P_n} = 0$ for the nth Qrr removed. The steady state concentration of *qrr* and HapR is computed in each strain then normalized by their corresponding wild-type values. The data (left), results from the model (middle), and corresponding error (right) are shown in Figure 3.6. The results are in good qualitative and quantitative agreement with the data.

3.2.2.3 Dosage Compensation and Qrr Feedback

To show that regulation in the sRNA circuit is responsible for dosage compensation, Svenningsen et al. measured luminescence in a wild-type, $\Delta hapR$, and in a *luxOAUCC* strain (a strain lacking the LuxO-Qrr feedback) with and then without all Qrr. Assuming the stability of each *qrr-lux* construct is the same and that fluorescence is proportional to the concentration of *qrr*, their data are normalized by the fluorescence from the *qrr1-lux* construct in the $\Delta hapR\Delta qrr1-4$ strain. The data, in Figure 3.7, show that removing one

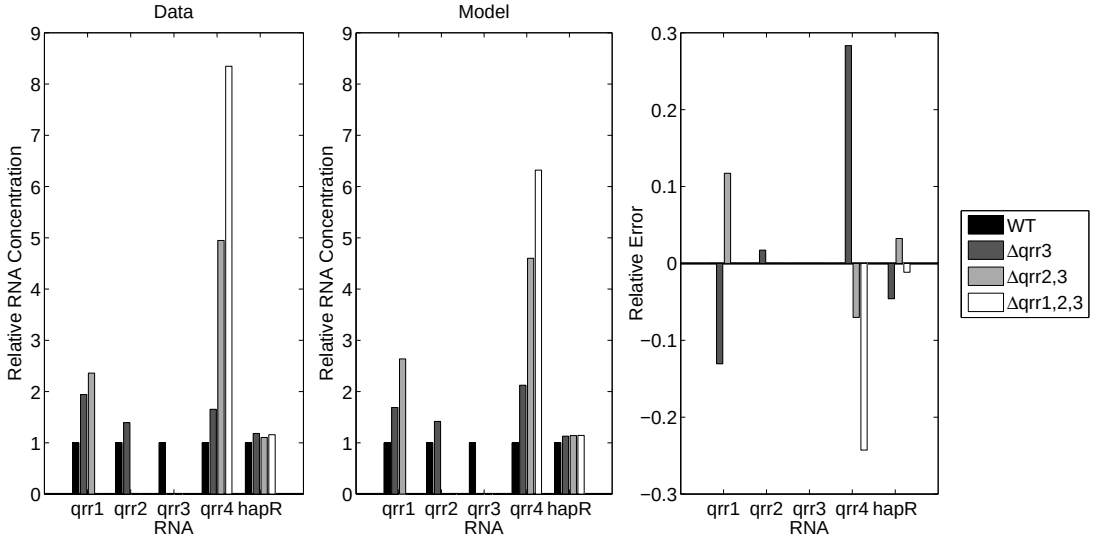

Figure 3.6. Comparison of the dosage compensation response for each Qrr between the data [94] (left) and the results (middle). Expression of each *qrr* increases relative to their wild-type concentrations at LCD when Qrr are sequentially removed. The error (right) shows that the model agrees well with the data overall.

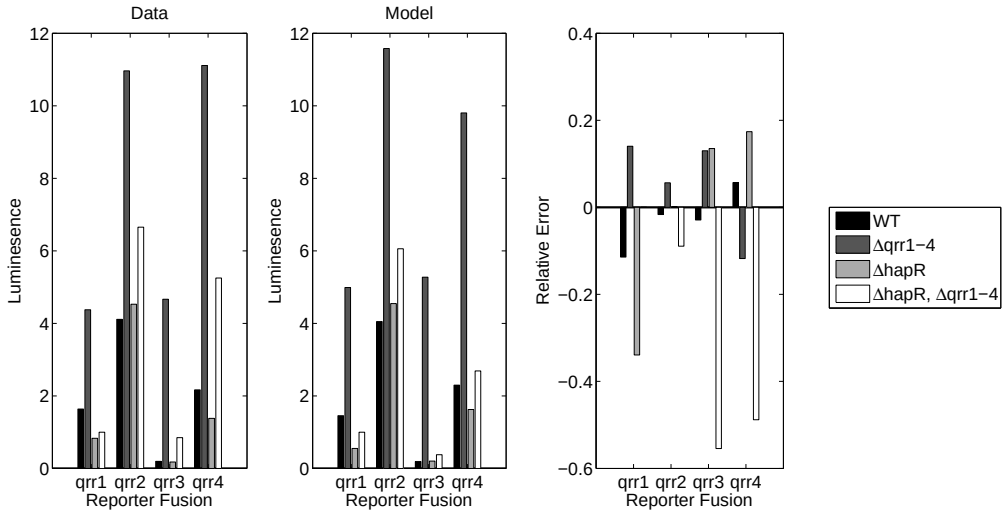

Figure 3.7. Comparison between the data [94] (left) and the results (middle) showing the fold change in *qrr-lux* luminescence in strains with and/or without Qrr and/or HapR. Luminescence from each construct is presented relative to *qrr1-lux* luminescence in the $\Delta hapR$ strain. The error (right) shows that the results agree well with the data.

or more *qrr* increases expression of the remaining *qrr*.

To model this experiment, a parameterization of each strain was created from the wild-type strain. For the $\Delta hapR$ strain, $E_{r_n} = \widehat{K}_{L_n} = 0$ for all n and, for the *luxOAUCC* strain, $E_{o_n} = V_{or} = 0$ for all n. Removing Qrr from these strains involved setting $\widehat{K}_{P_n} = 0$ for all n. The model of each *qrr-lux* construct is identical to the model of the Qrr promoter in (2.50), i.e.,

$$C_n(r,o) \quad = \quad \frac{\widehat{K}_{P_n}\Gamma o}{1+\widehat{K}_{P_n}\Gamma o}\frac{1+V_{q_n}(\widehat{K}_{L_n}r)^2}{1+(\widehat{K}_{L_n}r)^2}. \tag{3.2}$$

The steady state concentration of r and o in each strain is computed with and then without Qrr then used to evaluate (3.2). The luminescence from each promoter is normalized by its corresponding luminescence from the wild-type promoter. The results, Figure 3.7 (middle), show that the model agrees well with the data both qualitatively and quantitatively.

As an extension to the above experiment, Svenningsen et al. created a strain without the LuxO-Qrr feedback and examined the fold change in *qrr-lux* luminescence in a strain with vs. without Qrr. Their results, in Figure 3.8 (left), show that *qrr3* is most sensitive to changes in *qrr* whereas there is a more modest change in the remaining *qrr*.

To model this experiment, the wild-type strain parameterization was modified so that $E_{o_n} = 0$ for all n to remove the LuxO-Qrr feedback. The steady state concentrations of

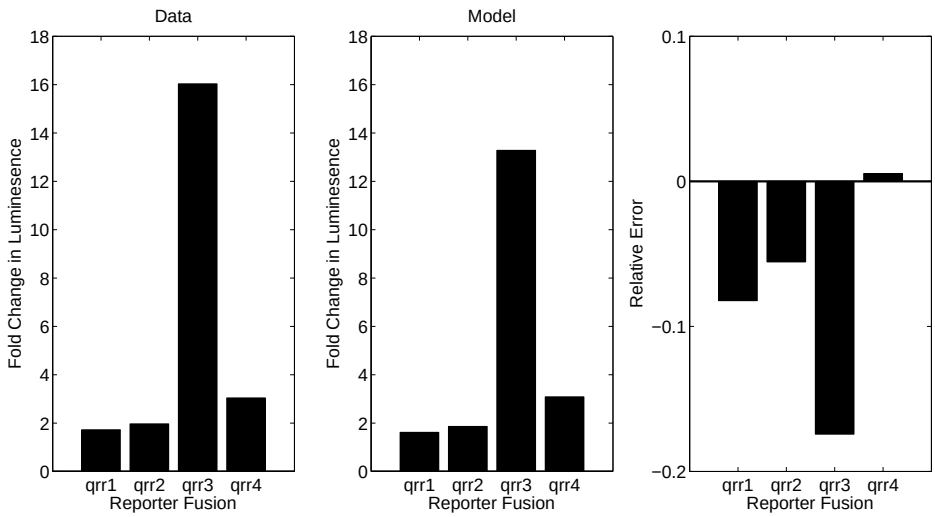

Figure 3.8. Comparison between the data [94] (left) and the results (middle) showing the fold change in *qrr-lux* luminescence when Qrr are removed in a strain without the LuxO-Qrr feedback. The error (right) shows that the model agrees well with the data.

HapR and LuxO in a strain with and then without *qrr* were then found. These steady-states were used to evaluate (3.2) to, again, determine the luminescence from each *qrr-lux* construct. Lastly, the luminescence of each $qrr-lux$ in the $\Delta qrr1-4$ strain were normalized by the luminescence in the strain with all *qrr*. Figure 3.8 shows that the model (middle) agrees well with the data (left).

3.2.3 Parameter Uncertainty, Identifiability, and Robustness

To understand what parameters are reliably estimated from the experiments by the model, the linearization of $\boldsymbol{F}(\boldsymbol{p})$ was studied at the parameterization determined in the previous section. If $||\delta\boldsymbol{p}||$ is small, then $\boldsymbol{F}(\boldsymbol{p}+\delta\boldsymbol{p}) = \boldsymbol{F}(\boldsymbol{p}) + D\boldsymbol{F}(\boldsymbol{p})\delta\boldsymbol{p} + O(||\delta\boldsymbol{p}||^2)$. Therefore, if each element in the column of $D\boldsymbol{F}(\boldsymbol{p})$ corresponding to parameter p_j is small (i.e., $\left|\frac{\partial F_i(\boldsymbol{p})}{\partial p_j}\right| \ll 1$ for all i), then the data are not very sensitive to p_j and, hence, the parameters are not estimated reliably.

We found that $\boldsymbol{F}(\boldsymbol{p})$ and $D\boldsymbol{F}(\boldsymbol{p})$ are accurate up to an order of 10^{-10}. Evaluating $\boldsymbol{F}(\boldsymbol{p})$ and $D\boldsymbol{F}(\boldsymbol{p})$ involves solving for the steady–states with a nonlinear solver that starts with a random initial guess. Hence, even with the same parameterization $\boldsymbol{p}$, the values of the forward map and its Jacobian may be slightly different from one simulation to another. We evaluated $\boldsymbol{F}(\boldsymbol{p})$ and $D\boldsymbol{F}(\boldsymbol{p})$ multiple times using the same parameterization and found that they differ up to 10^{-10} element-wise. Therefore, the jth parameter is assumed to be

a stationary solution of $\boldsymbol{F}(\boldsymbol{p}) = \boldsymbol{d}$ if $\left|\frac{\partial F_i(\boldsymbol{p})}{\partial p_j}\right| \leq 10^{-9}$ for all i. Figure 3.9 shows that the column norms of the Jacobian have at least one element greater than 10^{-9}. This suggests that all of the parameters could be identified using the data.

This also shows that *V. cholerae* parameters are more easily distinguished than the *V. harveyi* parameters and that *V. harveyi* E_{r_1} is the hardest parameter to distinguish in the data. This may explain why E_{r_1} is around 10^2-fold larger than the other E_{r_n} in *V. harveyi*. The result is somewhat expected given that the bulk of the *V. harveyi* data are very similar (i.e., showing LuxR-mCherry fluorescence as a function of autoinducer) rather than measurements from a variety of mutant strains.

To understand the parameter identification further, the singular value decomposition of the Jacobian matrix was computed to see what linear combination of parameters was associated with the smallest singular values and, hence, the weak search directions. Overall, the parameters associated with the smallest column norms of the Jacobian are also the main components of the right-singular vectors associated with the smallest singular values (see Table 3.2). These results again show that $E_{r_1}, E_{r_2}, E_{r_4}$, and E_{o_3} are difficult to identify in the *V. harveyi* data whereas E_{o_1}, E_{o_3}, E_{o_4}, E_{q_3}, E_{q_4}, E_{r_2}, E_{r_3}, Γ , $\widehat{K}_{p_3}$, and V_{q_2} are difficult to identify for *V. cholerae*. These results show that both the *V. harveyi* and *V. cholerae* parameterizations will benefit from new experiments that target these specific parameters.

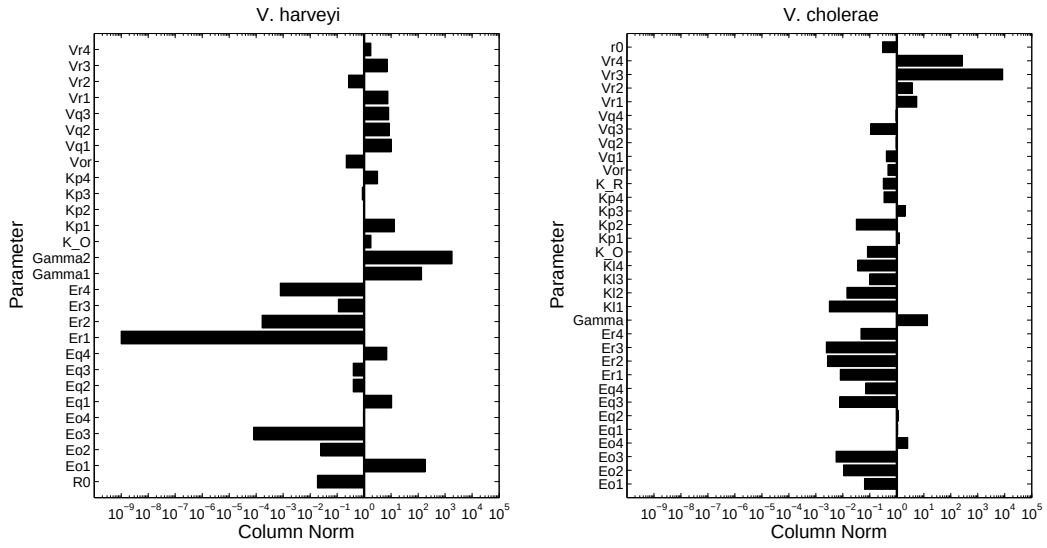

Figure 3.9. Comparison of the column norms of $D\boldsymbol{F}(\boldsymbol{p})$ for each parameter in *V. harveyi* and *V. cholerae*. The parameter corresponding to the column of the Jacobian is shown on the vertical axis, while the norm of the column is on the horizontal axis.

Table 3.2. Coefficients of the right-singular vectors associated with the smallest singular values of the Jacobian for *V. harveyi* and *V. cholerae* that are 10^{-7} smaller than the largest singular value. Coefficients smaller than 0.1 in absolute value are ignored.

V. harveyi

	σ_{n-3}	σ_{n-2}	σ_{n-1}	σ_n
E_{o_3}	1.0	–	–	–
E_{r_1}	–	–	–	−1.0
E_{r_2}	–	−0.1	1.0	–
E_{r_4}	–	−1.0	−0.1	–

V. cholerae

	σ_{n-7}	σ_{n-6}	σ_{n-5}	σ_{n-4}	σ_{n-3}	σ_{n-2}	σ_{n-1}	σ_n
E_{o_1}	–	−0.2	−0.5	−0.9	–	–	–	–
E_{o_3}	−0.1	1.0	−0.3	–	–	–	–	–
E_{o_4}	–	–	–	–	1.0	–	–	–
E_{q_3}	–	–	0.1	–	–	–	–	–
E_{q_4}	–	0.2	0.8	−0.5	–	–	–	–
E_{r_2}	–	–	–	–	–	–	–	−1.0
E_{r_3}	–	–	–	–	–	−0.1	1.0	–
Γ	–	–	–	–	–	−1.0	−0.1	–
K_{p_3}	−1.0	−0.1	–	–	–	–	–	–
V_{q_2}	–	–	−0.1	–	–	–	–	–

Up to now, only the Jacobian of the forward map has been used to discuss the uncertainty of the parameter estimates. To see how the parameter estimates are affected by non-linearities in the parameter estimation, 250 different realizations of the data were generated by randomly perturbing the data by at most 10% with a uniformly distributed random number. The model was then parameterized to each realization of the data by solving the problem in (3.1) and using the parameterization of each species in Table 3.1 as the initial estimate. The standard deviation of each parameter was then divided by its corresponding value in Table 3.1. The results are presented in Figure 3.10 and show that most of the parameters change on an order similar to the order of the error in the data. Therefore, with the exception of a few parameters, the *parameter estimation* for both *V. harveyi* and *V. cholerae* is robust in the sense that errors in the data give similar parameters. A Bayesian estimation (Monte Carlo analysis) is a more rigorous analysis of the uncertainty of parameter estimates. However, the cost of evaluating the forward problem and the number of realizations typically necessary for a Bayesian analysis made this analysis prohibitive, so it was not carried out here. Therefore, some parameters in the model cannot be reliably estimated from the experimental data considered. However, new experiments could be designed to specifically target these unresolved parameters and complete the model.

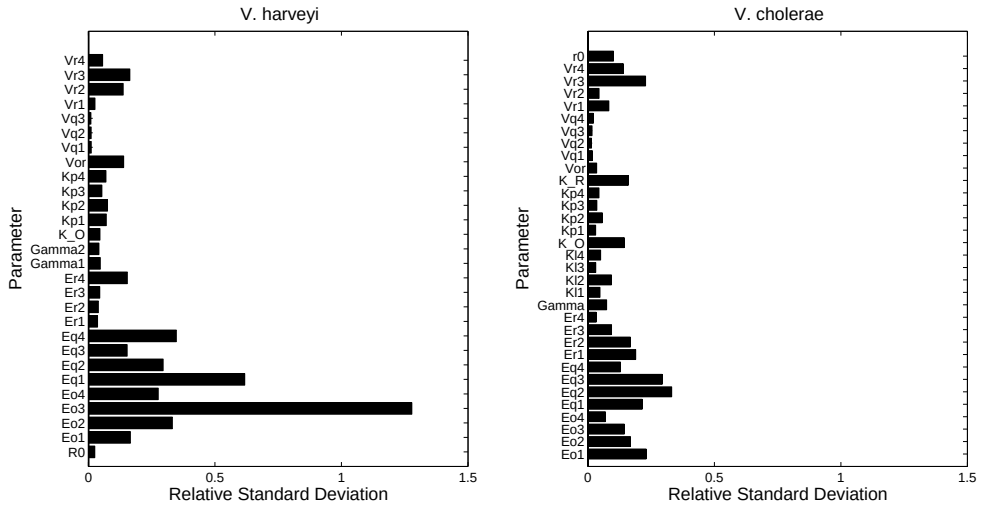

Figure 3.10. The standard deviation of each parameter in *V. harveyi* (left) and *V. cholerae* (right) relative to its corresponding value in Table 3.1. Results are based on 250 synthetic data generated by randomly perturbing the empirical data by at most 10%.

3.2.4 Species Comparisons and Qualitative Predictions

Although the sRNA circuits in *V. harveyi* and *V. cholerae* are topologically equivalent, the parameterization for each species is different. Here, the model is used to consider a series of experiments designed to identify qualitative differences in the responses of *V. harveyi* and *V. cholerae* and to understand the mechanisms responsible for these differences. The results show that abundance of Hfq-Qrr and changes to LuxO via the LuxO-Qrr feedback drive changes in *V. cholerae* Qrr concentration at LCD. Conversely, Hfq-Qrr is less abundant and Qrr less sensitive to changes in target mRNA in *V. harveyi*. Hence, dosage compensation is stronger in *V. cholerae* than in *V. harveyi* and that HapR is less sensitive than LuxR to changes in Qrr.

In what follows, the fold change of *qrr4* concentration is compared to the fold change in *qrr4* promoter activity between various strains. The concentration of *qrr4* is measured by modeling a real-time PCR analysis experiment and measure *qrr4* promoter activity by modeling the luminescence from a *qrr4-lux* construct. If the fold change in *qrr4* concentration is similar to the fold change in *qrr4-lux* luminescence, then the change in *qrr4* concentration is driven by a change in its expression rather than a change in its degradation via Hfq. Therefore, by comparing the fold change in *qrr4* concentration with the fold change in *qrr4* promoter activity, the degree to which changes in Hfq affect *qrr4* levels can be understood.

First, the the wild-type parameterization is modified to create parameterizations for

three different mutant strains: ΔLuxO-Qrr feedback ($E_{o_n} = 0$), ΔQrr feedback ($E_{o_n} = \widehat{K}_{L_n} = 0$), and ΔQrr feedback −LuxR/HapR ($E_{o_n} = E_{r_n} = \widehat{K}_{L_n} = 0$). For each strain including the wild-type strain, the steady state concentration of *qrr4*, *luxR/hapR*, and *luxO* at LCD ($\Gamma = \Gamma_{LCD}$) are computed. The steady state concentration of *luxR/hapR* and *luxO* along with (3.2) is used to measure the *qrr4-lux* luminescence for that particular strain. The *qrr4-lux* construct has the same mutations as the mutant strain (i.e., the model of the *qrr4-lux* construct for the ΔQrr feedback strain has $\widehat{K}_{L_n} = 0$). Lastly, the fold change in *qrr4* concentration and in *qrr4-lux* luminescence is compared between the strains indicated in Table 3.3. The results identify how Hfq and Qrr feedback regulate the concentration of *qrr4* in *V. harveyi* and *V. cholerae* at LCD.

The first row of Table 3.3 shows that addition of LuxR/HapR decreases *qrr4* in *V. cholerae* more than in *V. harveyi* because Hfq-Qrr is less abundant in *V. cholerae* than in *V. harveyi* in the absence of Qrr feedback. Note that *qrr4-lux* luminescence is constant because Qrr feedback is absent from both strains, so the change in *qrr4* concentration is driven by the change in Hfq-Qrr. In a ΔQrr feedback −LuxR/HapR strain, there is no target mRNA for Qrr to repress, so all available Hfq is bound by Qrr (i.e., $\sum_{n=1}^{4} H_n = 1$). On reintroducing LuxR/HapR, Qrr unbinds Hfq to repress LuxR/HapR. This diminishes the concentration of Hfq-Qrr and *qrr4* because more Hfq is available for it to bind. Therefore, Hfq-Qrr is less abundant in *V. cholerae* than in *V. harveyi* in the absence of Qrr feedback because *qrr4* decreases more in *V. cholerae* than in *V. harveyi*.

The second row of Table 3.3 shows that *qrr4* increases when the LuxR/HapR-Qrr feedback is reintroduced because LuxR/HapR enhances *qrr4* expression. This also shows that the concentration of Hfq-Qrr in the ΔLuxO-Qrr feedback strain is similar to that in a ΔQrr feedback strain because the change in *qrr4-lux* luminescence is similar to the

Table 3.3. Activation at the *qrr* promoter and Hfq determines the fold change in *qrr*4 expression. Differences between the fold change of *qrr4* concentration compared to the fold change in *qrr4-lux* luminescence shows the degree to which a change in *qrr4* concentration is driven by a change in *qrr*4 activation vs. Hfq.

		qrr4 Concentration		*qrr4-lux* Luminescence	
Expression in...	Relative to...	*V. harveyi*	*V. cholerae*	*V. harveyi*	*V. cholerae*
ΔQrr feedback	ΔQrr feedback −LuxR/HapR	0.88	0.75	1.00	1.00
ΔLuxO-Qrr feedback	ΔQrr feedback	1.61	1.53	1.59	1.19
WT	ΔLuxO-Qrr feedback	0.86	0.042	0.89	0.71

change in *qrr4* concentration. Therefore, although *qrr4* increases more in *V. harveyi* than in *V. cholerae*, *V. cholerae qrr4* remain less abundant than *V. harveyi qrr4*.

The last row of Table 3.3 shows that *qrr4* decreases more in *V. cholerae* than in *V. harveyi* when the LuxO-Qrr feedback is reintroduced because *V. cholerae qrr*4 is more sensitive to changes in LuxO. The LuxO-Qrr feedback represses LuxO, so the *qrr4* concentration should decrease as it does for both species. The results show that the change in $qrr4 - lux$ luminescence is comparable between the two species, so the changes in *qrr*4 concentration arise from differences in the sensitivity of their respective *qrr*4 promoter to changes in LuxO. Given the significantly greater decrease in *V. cholerae qrr*4 relative to the *V. harveyi qrr*4 concentration, hence, *V. cholerae qrr*4 is more sensitive to changes in LuxO than *V. harveyi qrr*4. The fold changes in *qrr*4 concentration between the second and third rows of Table 3.3 indicate that *V. harveyi* and *V. cholerae* are approximately equally sensitive to changes in LuxR/HapR whereas *V. cholerae qrr*4 is significantly more sensitive than *V. harveyi qrr*4 to changes in LuxO. These results were similar across all Qrr in *V. harveyi* and *V. cholerae*.

The above results suggest that dosage compensation is driven by changes in LuxO only in *V. cholerae* and by changes in LuxR and/or LuxO in *V. harveyi*. To test this, the fold change in *qrr*4, *luxR*/*hapR*, and *luxO* in *V. harveyi* and *V. cholerae* in a wild-type strain relative to a $\Delta qrr1-3$ strain are measured. As expected, *qrr*4 concentration increases in the $\Delta qrr1-3$ strain for both species (see Table 3.4). The results show that *luxR* and *luxO* increase significantly, but *hapR* increases only marginally. These results reflect the different sensitivities of the Qrr promoter to target mRNA. Dosage compensation of Qrr (i.e., the fold change in *qrr*4) arises when the expression of Qrr is sensitive to changes in target mRNA and when target mRNA is sensitive to changes in Qrr via Hfq. Given that *V. cholerae qrr*4 is significantly more sensitive to changes in LuxO than HapR, dosage compensation in *V. cholerae* is primarily driven by changes in LuxO. Similarly, *V. harveyi* Qrr are sensitive to changes in both LuxR and LuxO, so dosage compensation in *V. harveyi* is driven by changes in LuxR and LuxO.

Table 3.4. Fold change in *qrr4*, LuxR/HapR, and LuxO in a $\Delta qrr1-3$ strain relative to a wild-type strain at LCD for *V. harveyi* and *V. cholerae*.

Species	Fold Change in...		
	*qrr*4	*luxR*/*hapR*	*luxO*
V. harveyi	1.71	5.25	2.55
V. cholerae	6.32	1.14	3.16

3.3 Conclusion

In this work, the parameters for the mathematical model of the *V. harveyi* and *V. cholerae* sRNA circuit from Chapter 2 are estimated to explain why HapR is more robust than LuxR to changes in Qrr [57, 102]. The behavior of the model is consistent with a variety of empirical data from *V. harveyi* and *V. cholerae* sRNA. The model was parameterized by solving a nonlinear least-squares problem and identified most of the parameters from the data. The overall reliability of the parameter estimates and correspondence between the behavior of the model and the data implies that the current understanding of the biology is sufficient to explain a wide variety of behavior in *V. harveyi* and *V. cholerae*.

The similarities between the *V. harveyi* and *V. cholerae* quorum sensing circuits make it difficult to identify experimentally how and why their responses differ, yet it is much easier to do with the model. For example, an experiment that measures the difference in the *qrr* concentration and the luminescence from a *qrr-lux* construct between a wild-type strain and strains without one or both types of the Qrr feedback. This will determine the extent to which *qrr* levels change from dosage compensation compared to changes in Hfq. The results suggest that dosage compensation is driven by LuxO in *V. cholerae*, but by LuxO and LuxR in *V. harveyi*. This, in turn, may explain why *hapR*, rather than *luxR*, expression is more robust to changes in Qrr.

This same principle can help reduce the uncertainty in the estimates of the parameters. For example, there is little benefit estimating the parameters with new realizations of the same experiments, so new data from different experiments is needed. In particular, the SVD of the linearized forward map representing the experiments can be used to identify the set of experiments that reduce the dimension of its null space (for similar discussions see [14, 4, 6, 107]). The right-singular vectors associated with the smallest singular values identify the parameters that span the null space of the linearized forward map. The corresponding left-singular vectors, however, identify the measurements that yield little information about the parameters in the null space.

This work supports the hypothesis that the *V. harveyi* and *V. cholerae* quorum sensing circuits are topologically equivalent, yet tuned differently to elicit different responses [77]. Furthermore, these results show how *V. harveyi* and *V. cholerae* are tuned differently. To our knowledge, this is the first detailed model of the *V. harveyi* and *V. cholerae* sRNA circuits with physiologically-based estimates of the parameters. As such, the parameters can be used in similar models and the model can help design future experiments.

CHAPTER 4

MECHANISMS UNDERLYING THE ADDITIVE AND REDUNDANT QRR PHENOTYPES IN *VIBRIO HARVEYI* AND *VIBRIO CHOLERAE*

The quorum sensing systems of *V. harveyi* and *V. cholerae* are homologous and topologically similar, yet they respond differently to the same experimental conditions. In particular, *V. harveyi* Qrr are additive because all of its Qrr are required to maintain wild-type-like repression of LuxR whereas *V. cholerae* Qrr are redundant because any of its Qrr are sufficient to repress HapR. Given the striking similarities between their quorum sensing systems, experimentalists have been unable to identify conclusively the mechanisms behind these phenotypic differences. Nevertheless, the current hypothesis in the literature is that dosage compensation is the mechanism underlying redundancy.

In this chapter, the mechanisms underlying Qrr redundancy are studied using the model of the *V. harveyi* and *V. cholerae* sRNA circuit in Chapter 2 and the corresponding parameter estimates from Chapter 3. There are exactly two different cases underlying Qrr redundancy and that dosage compensation is unnecessary and insufficient to explain Qrr redundancy. Although *V. harveyi* Qrr are additive when the perturbations in Qrr are large, the model shows that *V. harveyi* and *V. cholerae* Qrr are redundant when the perturbations in Qrr are small. Hence, the additive and redundant Qrr phenotypes can emerge from parametric differences in the sRNA circuit. In particular, the affinity of Qrr and its expression relative to the master transcriptional regulator determine the level of redundancy in *V. harveyi* and *V. cholerae*. Furthermore, the results show that the additive and redundant Qrr phenotypes reflect differences in the concentration of Hfq-Qrr in *V. harveyi* and *V. cholerae*. The model is used to test the dosage compensation hypothesis with our alternative hypothesis and shows that decreasing the expression of

qrr, rather than removing dosage compensation, abolishes Qrr redundancy in *V. cholerae*. Further experimentation is needed to validate these results and test both Qrr redundancy hypotheses.

4.1 Introduction

Experiments show that *V. harveyi* Qrr are additive and *V. cholerae* Qrr are redundant [57, 102, 94]. The mechanistic reason for these phenotypic differences is not immediately apparent given the homology of their quorum sensing components and topology of their quorum sensing systems. Recently, experimentalists have shown that *qrr* expression increases via the Qrr feedback to compensate for the loss of Qrr and called this phenomenon dosage compensation [94]. They proposed, therefore, that differences in dosage compensation between *V. harveyi* and *V. cholerae* could explain the additive and redundant Qrr phenotypes. To understand how, note that removing one or more Qrr species partially derepresses target mRNA. This then increases expression of the remaining Qrr via the Qrr Feedback. They suggest that Qrr expression increases enough to offset the derepression of HapR in *V. cholerae*, so its Qrr are redundant, but not the derepression of LuxR in *V. harveyi*, so its Qrr are additive [94]. Recent studies mention their hypothesis, yet it remains untested [8, 33, 85, 91, 13].

In this chapter, the mechanisms underlying Qrr redundancy are identified then compared to the Qrr redundancy hypothesis in the literature. Analysis of the model shows that there are two different sets of criteria underlying Qrr redundancy. Although Qrr feedback contributes to redundancy in each case, the analysis suggests that Qrr can be redundant when one Qrr feedback is more dominant than the other rather than some synergetic relationship between the two. In general, however, Qrr feedback is unnecessary and insufficient for Qrr redundancy and dosage compensation diminishes when Qrr are redundant. Therefore, the results suggest that dosage compensation is not the mechanism underlying Qrr redundancy.

The parameterizations from Chapter 3 are then used to relate the results to *V. harveyi* and *V. cholerae*. First, the model independently produces qualitatively similar additive and redundant Qrr phenotypes for *V. harveyi* and *V. cholerae* to those in the literature. This suggests that parametric differences between the *V. harveyi* and *V. cholerae* sRNA circuit can explain the additive and redundant phenotypes. Surprisingly, the model shows that Qrr are redundant in both species when perturbations in Qrr are small. Additionally, the rate of *qrr* expression relative to the rate of $luxR/hapR$ expression and the affinity of Qrr to $luxR/hapR$ mRNA underly Qrr redundancy in both species. This implies that the

additive and redundant Qrr phenotypes reflect differences in the saturation of Hfq with Qrr.

Lastly, the dosage compensation Qrr redundancy hypothesis in the literature is contrasted with our alternate hypothesis. Changing the rate of *qrr* expression relative to *luxR*/*hapR* expression, for example, rather than removing the Qrr feedback, abolishes Qrr redundancy in *V. cholerae* and supports our alternate hypothesis. To our knowledge, this is the first to test the Qrr redundancy hypothesis of [94]. Additionally, the model lends itself well to testing what pathways in the *V. harveyi* and *V. cholerae* sRNA circuit contribute to the additive and redundant Qrr phenotypes.

In a similar study, [33] created a mathematical model of the *V. harveyi* and *V. cholerae* quorum sensing system using parameters that are guided by experiments to explain the additive and redundant Qrr phenotypes and other phenotypes up/downstream of the sRNA circuit [33]. They used a Gamma distribution to model the distribution of LuxR/HapR in a colony and showed that their model reproduces the phenotypic differences. They assumed that *V. harveyi* and *V. cholerae* only differ in the fold change of LuxR/HapR concentration necessary to activate luminescence and that, when cells enter stationary phase, environmental factors are necessary to increase HapR above a threshold to activate luminescence in *V. cholerae*. They also suggested that a decrease in Hfq during the stationary phase [46] could increase HapR over the desired threshold.

By contrast, our approach uses a detailed model of the *V. harveyi* and *V. cholerae* sRNA circuit with data-derived estimates of the parameters to model the expression of *luxR*/*hapR* as a function of LuxO-P:LuxO [45]. This approach has the advantage that the Qrr redundancy hypothesis from [94] can be tested and the specific differences between the *V. harveyi* and *V. cholerae* sRNA circuits underlying the additive and redundant Qrr phenotypes can be identified. Lastly, this approach shows a direct cause and effect relationship between Hfq and the additive and redundant Qrr phenotypes without additional assumptions about other regulatory pathways interacting with the sRNA circuit.

4.2 Results and Discussion

To discuss Qrr redundancy in the context of the model, a quantitative measure of Qrr redundancy that is consistent with the interpretation of experiments is needed. Qrr are redundant when there is relatively little change in *luxR*/*hapR* expression between a wild-type strain and an isogenic strain with one or more Qrr removed. Typically experimentalists change the concentration of *qrr* by deleting its gene. This corresponds to setting $\widehat{K}_{P_n} = 0$ in the model. However, the relative change in *luxR*/*hapR* expression scales with the relative change in *qrr* expression in the sense that the larger the change in *qrr* expression, the larger

the change in $luxR/hapR$ expression. To account for this, the relative change in $luxR/hapR$ expression is normalized by the relative change in qrr expression, i.e.,

$$\left| \frac{\Delta r}{r} \div \frac{\Delta \widehat{K}_{P_n}}{\widehat{K}_{P_n}} \right|. \tag{4.1}$$

[94] argued that wild-type-like repression of LuxR/HapR occurs only when the changes in Qrr are small, so, as $\Delta \widehat{K}_{P_n} \to 0$, the sensitivity becomes

$$\sigma_n \equiv \lim_{\Delta \widehat{K}_{P_n} \to 0} \left| \frac{\Delta r}{r} \div \frac{\Delta \widehat{K}_{P_n}}{\widehat{K}_{P_n}} \right| = \left| \frac{\widehat{K}_{P_n}}{r} \frac{\mathrm{d}r}{\mathrm{d}\widehat{K}_{P_n}} \right|. \tag{4.2}$$

Therefore, Qrr are redundant if LuxR/HapR is insensitive to changes in $\widehat{K}_{P_n}$.

Conversely, Qrr are additive when a small relative change in Qrr causes a large relative change in LuxR/HapR. Following a similar reasoning to that above, Qrr additivity corresponds to when the sensitivity of LuxR/HapR with respect to $\widehat{K}_{P_n}$ is large. Thus, the sensitivity of LuxR/HapR with respect to $\widehat{K}_{P_n}$ is a measure of the degree to which Qrr are redundant. Importantly, that this measure of Qrr redundancy is based on the *interpretation* of the additive and redundant Qrr data proposed by [57, 102, 94].

4.2.1 Steady State Analysis of the Simplified Model

In this section, the model is simplified to make its analysis more tractable and to illustrate the essential ideas underlying Qrr redundancy. Suppose that each Qrr degrades target mRNA equally (i.e., $E_{r_n} = E_r^*$, $E_{o_n} = E_o^*$) and that there is no autoregulation (i.e., $\widehat{K}_R = \widehat{K}_O = 0$). Throughout this work, the model with these simplifications is referred to as the *simplified model* and the model without these simplifications is referred to as the *full model*. With these simplifications, the steady state solution for r and o can be expressed in terms of $S = \sum_n H_n$:

$$r = \frac{1}{1 + E_r^* S} \qquad\qquad o = \frac{1}{1 + E_o^* S}. \tag{4.3}$$

Similarly, the steady state solution for q_n in terms of S is

$$q_n = \left(\frac{1}{1 + E_{q_n}(1 - S)} \right) \left(\frac{\widehat{K}_{P_n} \Gamma}{1 + E_o^* S + \widehat{K}_{P_n} \Gamma} \right) \left(\frac{(1 + E_r^* S)^2 + V_{q_n} \widehat{K}_{L_n}^2}{(1 + E_r^* S)^2 + \widehat{K}_{L_n}^2} \right). \tag{4.4}$$

Using (4.3) and (4.4), (2.51) simplifies to $0 = \Phi_n - D(S) H_n$ where

$$\Phi_n = \frac{1}{V_{r_n}} P_{O_n}(S) P_{R_n}(S) P_{H_n}(S) \tag{4.5}$$

$$D(S) = \frac{E_r^*}{1 + Er^* S} + V_{or}\frac{E_o^*}{1 + E_o^* S}, \tag{4.6}$$

$$P_{O_n}(S) = \frac{\widehat{K}_{P_n}\Gamma}{1 + E_o^* S + \widehat{K}_{P_n}\Gamma}, \tag{4.7}$$

$$P_{R_n}(S) = \frac{(1 + E_r^* S)^2 + V_{q_n}\widehat{K}_{L_n}^2}{(1 + E_r^* S)^2 + \widehat{K}_{L_n}^2}, \tag{4.8}$$

$$P_{H_n}(S) = \frac{E_{q_n}(1 - S)}{1 + E_{q_n}(1 - S)}. \tag{4.9}$$

Summing over all n gives the following equation for the steady state solution of S:

$$0 = \Sigma_n(S) - D(S)S, \tag{4.10}$$

where

$$\sum_{n=1}^{N} \Phi_n \equiv \Sigma_n(S). \tag{4.11}$$

Given that the solution of r, o, q_n, and H_n are expressed in terms of S, understanding the steady state behavior of S is sufficient to understand the steady state behavior of the system as a whole. To this end, note that $D(S)S$ is an increasing, concave down function that is zero at $S = 0$ and approaches $1 + V_{or}$ for large S. Φ_n is the product of three decreasing functions of S.

The first function, $P_{O_n}(S)$, represents the expression of *qrr* from LuxO-P and is a decreasing, concave up function of S that is bounded below by 0 and above by $\widehat{K}_{P_n}\Gamma/(1 + \widehat{K}_{P_n}\Gamma) < 1$. Additionally, $P_{O_n}(S)$ is an increasing, concave down function of $\widehat{K}_{P_n}\Gamma$. The second function in Φ_n, $P_{R_n}(S)$, represents the degree to which LuxR/HapR enhances *qrr* expression. This is a decreasing, sigmoidal function of S that is bounded below by 1 and above by V_{q_n} (note that $V_{q_n} \geq 1$ because LuxR/HapR enhances Qrr expression). The amplitude of $P_{R_n}(S)$ is determined by V_{r_n}, whereas the transition between its extremes occurs when $\widehat{K}_{L_n} \approx 1 + E_r^* S$. This function increases the amplitude of Φ_n when $S \approx 0$ and has little effect on Φ_n when $\widehat{K}_{L_n} \ll 1 + E_r^* S$.

The last function, $P_{H_n}(S)$, reflects how the availability of Hfq limits repression of target mRNA. Similar to the previous two functions, $P_{H_n}(S)$ is a decreasing, concave down function of S that is bounded below by 0 and below by $E_{q_n}/(1 + E_{q_n})$. Importantly, $P_{H_n}(S)$ is responsible for establishing an upper bound on the solution of S because $P_{H_n}(S) \leq 0$ for $S \geq 1$. Figure 4.1 shows an example of each function in Φ_n (top) and the intersection of $D(S)S = \Sigma_n(S)$ (bottom).

There is only one steady state solution for S because $D(S)S$ and $\Sigma_n(S)$ are, respectively, increasing and decreasing functions of S that intersect only once. We know that $D(S)S$

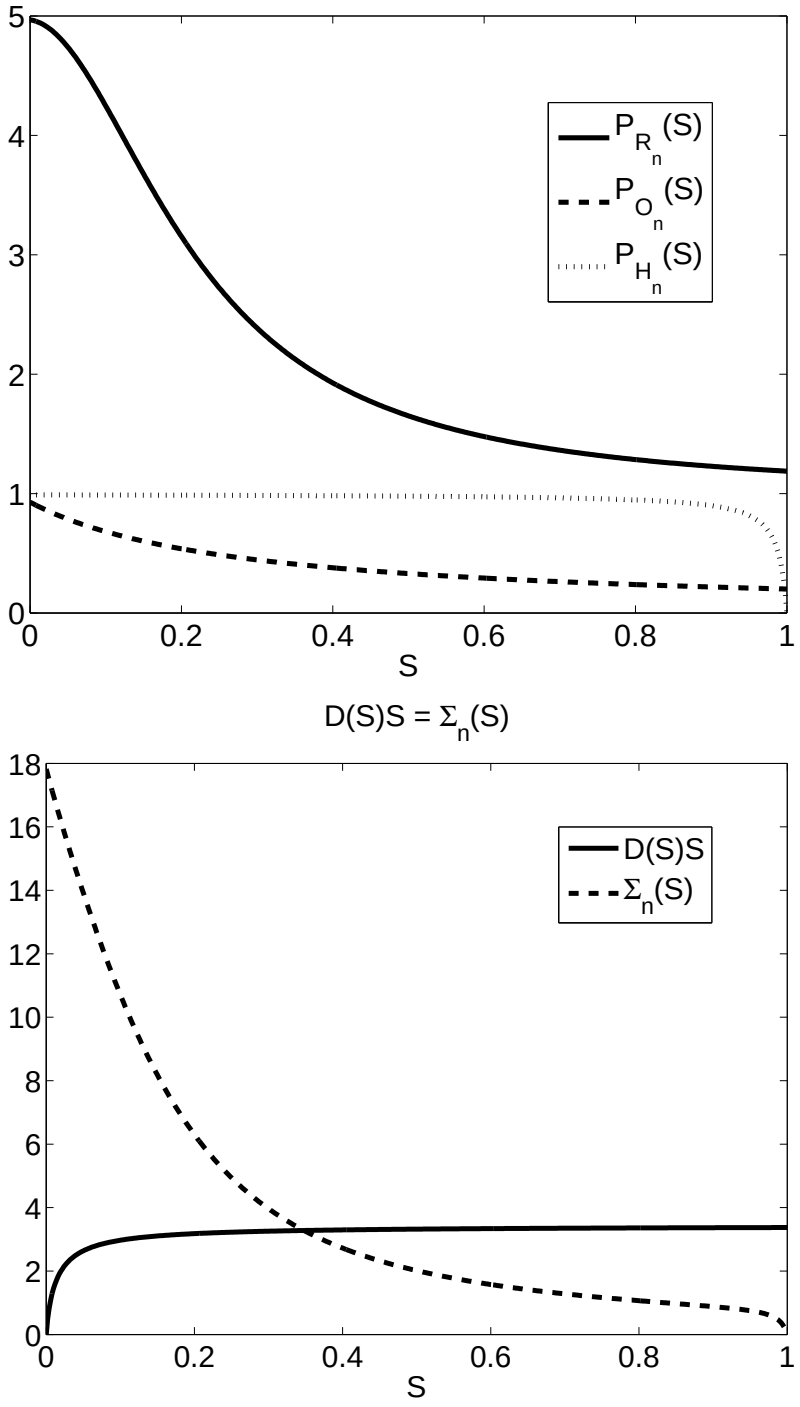

Figure 4.1. General structure of the steady state solution of S in the simplified model. The general qualitative structure of the functions in Φ_n are shown on the top and the representative nullclines of $\Sigma_n(S)$ and $D(S)S$ are shown on the bottom. Parameters are chosen to highlight the qualitative features of the each curve.

starts at the origin and increases with S whereas $\Sigma_n(S)$ starts above the origin and decreases to 0 when $S = 1$. Therefore, these curves intersect exactly once between $0 \leq S < 1$. Furthermore, given that $\Sigma_n(S) > D(S)S = 0$ at $S = 0$ and $0 = \Sigma_n(S) < D(S)S$ at $S = 1$, the steady state is stable. This result is consistent with empirical data that show there is a single, graded transition to/from the LCD and HCD states [57, 102, 62, 99]. This type of stability differs from LuxIR-type quorum sensing systems. These latter systems are bistable [24] and, therefore, have two different critical cell-population densities for the LCD to HCD transition [116, 117].

4.2.2 An Overview of Redundancy

The plots of $D(S)S$ and $\Sigma_n(S)$ in Figure 4.2 illustrate how the curves look when Qrr are redundant. In Figure 4.2 (top), the amplitude of $\Sigma_n(S)$ is significantly lower than that of $D(S)S$ and, most importantly, the curves intersect where $D(S)S$ is increasing rapidly. Therefore, although a change in $\widehat{K}_{P_n}$ moves $\Sigma_n(S)$ up/down along $D(S)S$, the equilibrium of S remains relatively constant. The other case of Qrr redundancy is illustrated in Figure 4.2 (bottom). The amplitudes of the curves are significantly different once again, but they intersect near $S = 1$ where $\Sigma_n(S)$ decreases rapidly. Changing $\widehat{K}_{P_n}$ moves $\Sigma_n(S)$ up/down (i.e., parallel to its slope), but the equilibrium value of S remains relatively constant.

Figure 4.2 shows that, as a consequence of Qrr redundancy, $D(S)S$ and $\Sigma_n(S)$ intersect one another when one of them is approximately vertical. Given that the curves are vertical near $S \approx 0, 1$, this means that there are two different criteria resulting in Qrr redundancy and each corresponds to different saturated levels of Hfq.

4.2.3 Qrr Are Redundant when $S \approx 0, 1$ Only

First we show that Qrr are redundant when $S \approx 0, 1$ then use the geometry of the intersection of $D(S)S$ with $\Sigma_n(S)$ to argue that these are the only cases of redundancy. Using (2.51) in the full model, $H_n \to 0$ as $V_{r_n} \to \infty$. In this limit, repression of r is independent of Qrr and $\sigma_n = 0$. Therefore, Qrr are redundant in this limit. Similarly, $\sum_n H_n \to 1$ as $V_{r_n} \to 0$ from (2.51). Using $\sum_n H_n = 1$ to eliminate H_4 in (2.48) results in the following steady state solution of r:

$$0 = r_0 + \frac{1 - r_0}{1 + (\widehat{K}_R r)^2} - \left(\sum_{n=1}^{3} (E_{r_n} - E_{r_4}) H_n + E_{r_4} + 1 \right) r. \tag{4.12}$$

If $E_{r_n} = E_r^*$ for all n, then r is independent of H_n, so $\sigma_n = 0$. Therefore, Qrr are redundant when $V_{r_n} \to \infty$ or as $V_{r_n} \to 0$ and $E_{r_n} \approx E_r^*$ for all n.

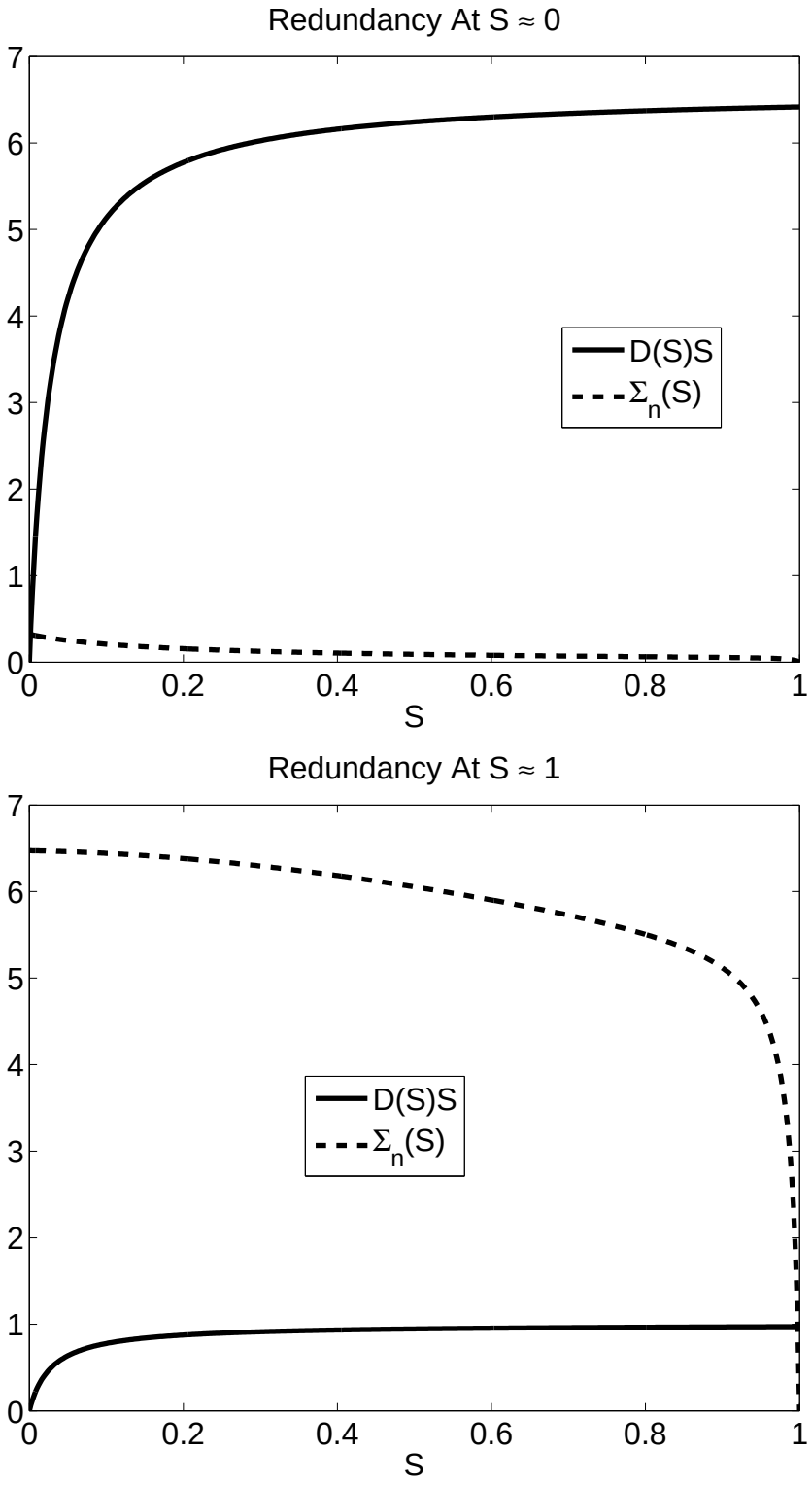

Figure 4.2. The intersection of $D(S)S$ with $\Sigma_n(S)$ when Qrr are redundant. Parameters are chosen manually to highlight the qualitative features of $D(S)S$ and $\Sigma_n(S)$.

The geometry of the intersection of $D(S)S$ with $\Sigma_n(S)$ of the simplified model imply that these are the only two cases where Qrr are redundant. First note that $D(S)S$ is independent of V_{r_n}, so consider how the intersection of $\Sigma_n(S)$ with $D(S)S$ changes as $V_{r_n} \to \infty$. In this limit, Qrr are redundant and $S \approx 0$. Based on (4.11), $\Sigma_n(S) \to 0$ as $V_{r_n} \to \infty$. Consequently, the curves intersect near $S \approx 0$ where the derivative of $D(S)S$ with respect to S is greatest. When $\widehat{K}_{P_n}$ changes, $\Sigma_n(S)$ moves up/down along $D(S)S$ and, because the slope of $D(S)S$ is steep, the equilibrium solution of S changes little. Similarly, as $V_{r_n} \to 0$ and all other parameters are fixed, $\Sigma_n(S) \gg D(S)S$ for $S < 1$. However, $\Sigma_n(S) \to 0$ as $S \to 1$ because $\Sigma_n(1) = 0$, regardless of V_{r_n}. Consequently, $D(S)S$ and $\Sigma_n(S)$ intersect near $S \approx 1$ where the slope of $\Sigma_n(S)$ is steepest. Therefore, their intersection is relatively unchanged when $\widehat{K}_{P_n}$ changes. In both cases, the slope of either $D(S)S$ or $\Sigma_n(S)$ is steep at its intersection, which limits the relative change in the equilibrium value of S when $\widehat{K}_{P_n}$ changes. Therefore, Qrr are redundant when the slope of either $D(S)S$ or $\Sigma_n(S)$ is steep at its intersection. Given that this only occurs when $S \approx 0, 1$, these are the only two solutions of S when Qrr are redundant.

4.2.4 General Qrr Redundancy Criteria when $S \approx 0$

In this section, specific criteria underling redundancy in the simplified model when $S \approx 0$ are identified by approximating the solution of S to estimate σ_n. Because $S \to 0^+$ as $V_{r_n} \to \infty$, the solution for S is assumed to be a power series of the form $S = \sum_n c_n^{(0)} V_{r_n}^{-1}$. This expression is substituted into $D(S)S = \Sigma_n(S)$ and $c_n^{(0)}$ solved for to first order in $V_{r_n}^{-1}$ to get

$$S = \frac{1}{D(0)} \sum_n \frac{P_{H_n}(0) P_{O_n}(0) P_{R_n}(0)}{V_{r_n}}. \tag{4.13}$$

This result is used to estimate the sensitivity when $S \approx 0$, $\sigma_n^{(0)}$. To this end, (4.13) is substituted into (4.2) to get

$$\sigma_n^{(0)} = \left(\frac{1}{1 + \widehat{K}_{P_n} \Gamma} \right) \frac{E_r^* c_n^{(0)}}{V_{r_n}}. \tag{4.14}$$

Note that, because the term in the parentheses in (4.14) is bounded above by 1 and that $E_r^* D(0)^{-1} \to 1$ as $E_r^* \to \infty$, the sensitivity is on the same order as $E_r^* c_n^{(0)} V_{r_n}^{-1}$. The approximation (4.13) and corresponding sensitivity estimate (4.14) also hold under the more general assumption that $c_n^{(0)} V_{r_n}^{-1} \ll 1$. Therefore, if $c_n^{(0)} V_{r_n}^{-1} \ll 1$, then Qrr are redundant.

Although $V_{r_n} \to \infty$ is sufficient for Qrr redundancy, other parameter ranges can result in redundancy as well. Rewriting (4.14) gives

$$\sigma_n^{(0)} = \frac{E_r^*}{E_r^* + V_{or}E_o^*} \cdot \frac{1 + V_{q_n}\widehat{K}_{L_n}^2}{1 + \widehat{K}_{L_n}^2} \cdot \frac{1}{V_{r_n}} \cdot \frac{\widehat{K}_{P_n}\Gamma}{(1 + \widehat{K}_{P_n}\Gamma)^2} \cdot \frac{E_{q_n}}{1 + E_{q_n}}. \tag{4.15}$$

This shows, for example, that the stronger the LuxO-Qrr feedback is relative to the LuxR/HapR-Qrr feedback ($V_{q_n} \approx 1$, $E_o^* \gg E_r^*$), the smaller the sensitivity. Furthermore, the sensitivity is also small when the rate of *qrr* expression is less than *luxR/hapR* expression ($V_{r_n} \gg 1$), the rate of *luxR/hapR* expression is less than *luxO* expression ($V_{or} \gg 1$), Qrr are unstable ($E_{q_n} \approx 0$), or when the binding affinity of LuxO-P to the *qrr* promoter is weak ($\widehat{K}_{P_n}\Gamma \ll 1$). Therefore, Qrr feedback is unnecessary nor sufficient for Qrr to be redundant in general because other mechanisms can diminish sufficiently the sensitivity.

4.2.5 General Qrr Redundancy Criteria when $S \approx 1$

The analysis above is repeated to estimate S when $S \approx 1$ and approximate the sensitivity in this case. Given that $S \to 1^-$ as $V_{r_n} \to 0$, the solution of S is approximated using a power series in terms of V_{r_n} of the form $S = 1 - \sum_{n=1}^{N} c_n^{(1)} V_{r_n}$ where $V_{r_n} \ll 1$ for all n. After substituting this expression into $D(S)S = \sum_{n=1}^{N} \Phi_n$, $c_n^{(1)}$ is solved for to get

$$S = 1 - D(1) \sum_{n=1}^{N} \frac{V_{r_n}}{E_{q_n} P_{O_n}(1) P_{R_n}(1)}. \tag{4.16}$$

This approximation is used to estimate the sensitivity when $S \approx 1$, $\sigma_n^{(1)}$,

$$\sigma_n^{(1)} = \left(\frac{E_r^*}{1 + E_r^*}\right)\left(\frac{1 + E_o^*}{1 + E_o^* + \widehat{K}_{P_n}\Gamma}\right) c_n^{(1)} V_{r_n}. \tag{4.17}$$

Each term in the parentheses in (4.17) is bounded above by 1, so $\sigma_n^{(1)}$ is on the same order as $c_n^{(1)} V_{r_n}$. As before, the sensitivity is small provided $c_n^{(1)} V_{r_n} \ll 1$, so a variety of criteria summarized by the relationship $c_n^{(1)} V_{r_n} \ll 1$ cause Qrr redundancy.

As before, there is no minimum sensitivity, so Qrr are redundant whenever the sensitivity is reduced. To understand this balance, note that (4.17) can be rewritten as

$$\sigma_n^{(1)} = \left((1 + E_o^*)\left(\frac{E_r^*}{1 + E_r^*}\right)^2 + V_{or}\frac{E_o^* E_r^*}{1 + E_r^*}\right) \frac{(1 + E_r^*)^2 + \widehat{K}_{L_n}^2}{(1 + E_r^*)^2 + V_{q_n}\widehat{K}_{L_n}^2} \cdot \frac{1}{E_{q_n}} \cdot \frac{V_{r_n}}{\widehat{K}_{P_n}\Gamma} \tag{4.18}$$

Similar to the previous case of redundancy, Qrr are redundant when the LuxR/HapR-Qrr feedback is stronger than the LuxO-Qrr feedback ($V_{q_n} \gg 1$, $E_o^* \ll 1$, and $\widehat{K}_{L_n} \gg 1 + E_r^*$). Additionally, other factors decrease the sensitivity such as when the rate of *qrr* expression

is greater than that of $luxR/hapR$ ($V_{r_n} \ll 1$), the rate of $luxO$ expression is less than $luxR/hapR$ expression ($V_{or} \ll 1$), Qrr are stable ($E_{q_n} \gg 1$), or when the binding affinity of LuxO-P to the qrr promoter is strong ($\widehat{K}_{P_n}\Gamma \gg 1$). Again, Qrr can be redundant in the absence of feedback provided that $V_{r_n}/(E_{q_n}\widehat{K}_{P_n}\Gamma) \ll 1$. Therefore, Qrr feedback is neither necessary nor sufficient for Qrr redundancy.

4.2.6 The Relationship Between Dosage Compensation and Qrr Redundancy

Svenningsen et al. sequentially removed Qrr in *V. cholerae* and showed that the Qrr feedback is necessary to increase the activity of the remaining qrr promoters. They called this phenomenon *dosage compensation*. Consequently, they also noticed that HapR repression remained relatively unchanged between a wild-type strain and mutant strains with one to three Qrr [94]. They, therefore, argued that qrr expression changed to compensate for the loss of other Qrr and maintain a similar level of HapR repression [94, 102, 57]. The results suggest that, not only is Qrr feedback unnecessary and insufficient for Qrr redundancy in general, but that dosage compensation diminishes as the sensitivity of LuxR/HapR with respect to Qrr diminishes.

Qrr feedback is unnecessary and insufficient for Qrr redundancy since the criterion for Qrr redundancy depends on a variety of mechanisms other than the Qrr feedback. The sensitivity is small when there is a strong bias for one Qrr feedback rather than some synergy between the two. Redundancy also depends on the binding affinity of LuxO-P to each qrr promoter ($\widehat{K}_{P_n}\Gamma$), the relative rates of expression of $luxR/hapR$ and qrr (V_{r_n}), and the stability of Qrr relative to its binding rate to Hfq (E_{q_n}). In other words, the sensitivity can be small even in the absence of Qrr feedback. Therefore, although a strong bias for one Qrr feedback can decrease the sensitivity, Qrr feedback is neither necessary nor sufficient for Qrr redundancy.

Dosage compensation decreases with the sensitivity. Dosage compensation is the relative fold change in the qrr promoter activity as the concentration of the other Qrr changes. The model of the qrr promoter activity is $Q_{P_n} = P_{O_n}(S)P_{R_n}(S)$ and, hence, measure of dosage compensation is

$$\frac{\mathrm{d}Q_{P_n}}{Q_{P_n}} = \left(\frac{1}{P_{O_n}(S)}\frac{\mathrm{d}P_{O_n}(S)}{\mathrm{d}S} + \frac{1}{P_{R_n}(S)} \cdot \frac{\mathrm{d}P_{R_n}(S)}{\mathrm{d}S}\right)\mathrm{d}S. \tag{4.19}$$

Based on the geometry of the intersection of $D(S)S$ and $\Sigma_n(S)$, along with the qualitative shape of $P_{O_n}(S)$ and $P_{R_n}(S)$, $\mathrm{d}S$ and the derivatives of $P_{O_n}(S)$ and $P_{R_n}(S)$ are small when Qrr are redundant. This implies that dosage compensation is small as well. This

agrees with our intuition on the relationship between dosage compensation and redundancy as well. Dosage compensation relies on the expression of *qrr* being sensitive to changes in target mRNA and on target mRNA being sensitive to changes in Qrr via Hfq. When Qrr are redundant, target mRNA remain relatively constant with changes in Qrr and the *qrr* promoter is saturated. Therefore, the lower the sensitivity, the weaker the dosage compensation response.

4.2.7 *V. harveyi* Qrr Are Redundant when Qrr Perturbations Are Small

Up to this point, the discussion has mainly focussed on identifying the mechanisms underlying Qrr redundancy in the simplified model. In this and the following sections, the mechanisms underlying the additive and redundant Qrr phenotypes in *V. harveyi* and *V. cholerae* are studied by analyzing the full model. First, the model with the *V. harveyi* and *V. cholerae* parameterizations are shown to qualitatively produces the additive and redundant Qrr phenotypes. This serves as validation of the model with independent data and, hence, as motivation to use the model for further study. Then $S = \sum_n H_n$ is measured to predict whether $S \approx 0$ or $S \approx 1$ when Qrr are redundant. This result leads to identifying 30 different parametric constraints that lead to Qrr redundancy.

To show that the model qualitatively reproduces the additive and redundant Qrr phenotypes in the data, the experiment is modeled using the model of the sRNA circuit along with the *V. harveyi* and *V. cholerae* parameterizations. To this end, the steady state concentration of LuxR/HapR is computed over a large range of Γ in a wild-type strain and in isogenic mutant strains that have only one Qrr. This protocol is almost identical to that used in the original studies showing that *V. harveyi* and *V. cholerae* Qrr were additive and redundant, respectively [57, 102]. The main difference is that we measure the steady state concentration of LuxR/HapR as a function of LuxO-P:LuxO rather than the bioluminescence as a function of time and cell-population density.

The results in Figure 4.3 show that the model qualitatively reproduces the additive and redundant Qrr phenotypes in *V. harveyi* and *V. cholerae*, respectively. In particular, HapR repression at LCD, as well as the rate and onset of the LCD to HCD transition, are similar between the *V. cholerae* strains but not between the *V. harveyi* strains. Importantly, the additive and redundant Qrr phenotype data were not used to parameterize the model in Chapter 3, so this result independently validates the model and its parameters. Altogether, this suggests that parametric differences in the *V. harveyi* and *V. cholerae* sRNA circuits can cause the additive and redundant Qrr phenotypes as suggested in [94].

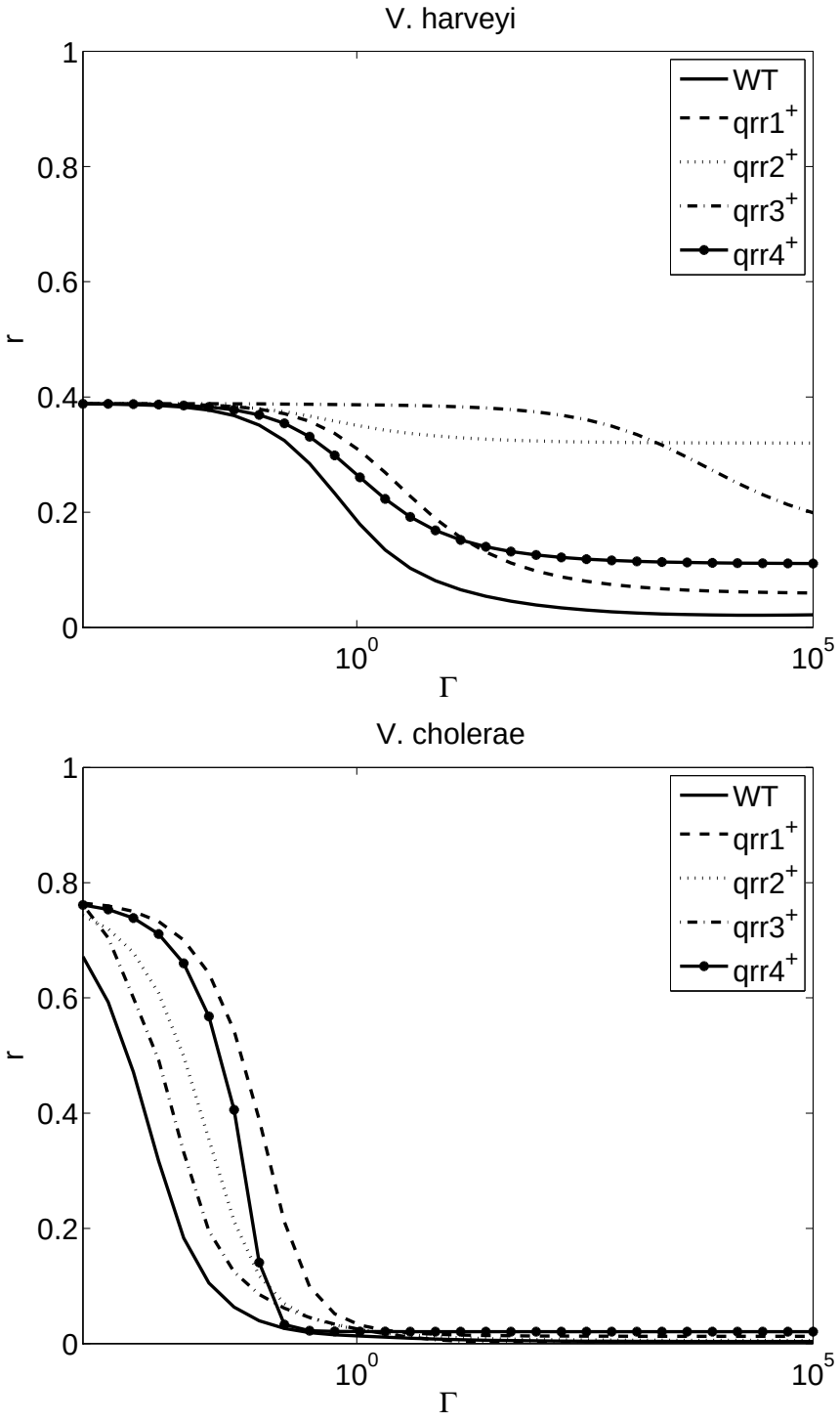

Figure 4.3. LuxR/HapR in a wild-type strain and isogenic strains with one Qrr over a range of Γ spanning LCD to HCD. The model qualitatively reproduces the additive and redundant Qrr phenotypes of the *V. harveyi* and *V. cholerae*, respectively.

The analysis of the simplified model showed that there are two different conditions that lead to Qrr redundancy. We set out to identify which case applies to *V. cholerae* and expected to show that the sensitivity is greater in *V. harveyi* than in *V. cholerae*. Based on Figure 4.3, $\Gamma = 10^5$ is assumed to be large enough to represent the LCD state for *V. harveyi* and *V. cholerae*. We compute σ_n for the full model at $\Gamma = 10^5$. The results in Table 4.1 show that $S \approx 0$ in *V. harveyi* and $S \approx 1$ in *V. cholerae* and that the sensitivity is comparatively small in both species, rather than in *V. cholerae* only. Even though *V. harveyi* Qrr are additive when the perturbations in Qrr are large (Figure 4.3, top), Table 4.1 suggests that *V. harveyi* Qrr are redundant when the perturbations in Qrr are small. Although these results are contrary to what was expected, they support the claim that the sRNA circuit can only compensate for small changes in Qrr [94]. These results are also consistent with the results of the simplified model for they show that $S \approx 0, 1$ when Qrr are redundant.

4.2.8 Mechanisms Underlying Redundancy in *V. harveyi* and *V. cholerae*

To identify why *V. cholerae* Qrr, but not *V. harveyi* Qrr, are redundant when the perturbations in Qrr are large, the reason(s) underlying Qrr redundancy when the perturbations in Qrr are small must be understood. Given that *V. harveyi* Qrr are redundant and $S \approx 0$ at LCD, then *V. harveyi* σ_2 should increase if $V_{q_2}, \widehat{K}_{L_2}, E_{r_2}$ or E_{q_2} are increased and should decrease if E_{o_2}, V_{or}, V_{r_2}, or $\widehat{K}_{P_2}\Gamma$ are increased based on (4.15). Similarly, given that *V. cholerae* Qrr are redundant and $S \approx 1$ at LCD, *V. cholerae* σ_3 should increase if V_{or}, V_{r_3}, E_{o_3}, or E_{r_3} are increased and decrease if $V_{q_3}, E_{q_3}, \widehat{K}_{L_3}$, or $\widehat{K}_{P_3}\Gamma$ are decreased following from (4.18). We chose σ_2 in *V. harveyi* because σ_2 is the second smallest sensitivity next to σ_3 and H_2 is approximately 10^2 smaller than H_3. We chose σ_3 in *V. cholerae* because

Table 4.1. Measurements of the sensitivities for each Qrr (σ_n), total concentration of bound Hfq (S), and the concentration of Hfq bound with each Qrr (H_n) in *V. harveyi* and *V. cholerae* at $\Gamma = 10^5$ using the parameters in Table 3.1.

	V. harveyi	*V. cholerae*
σ_1	$1.03{\cdot}10^{-1}$	$1.60{\cdot}10^{-4}$
σ_2	$8.52{\cdot}10^{-4}$	$7.48{\cdot}10^{-8}$
σ_3	$7.04{\cdot}10^{-4}$	$9.53{\cdot}10^{-3}$
σ_4	$1.35{\cdot}10^{-2}$	$1.05{\cdot}10^{-3}$
S	$9.95{\cdot}10^{-2}$	1.00
H_1	$5.32{\cdot}10^{-5}$	$4.87{\cdot}10^{-3}$
H_2	$1.17{\cdot}10^{-3}$	$4.77{\cdot}10^{-4}$
H_3	$8.80{\cdot}10^{-2}$	$9.07{\cdot}10^{-1}$
H_4	$1.02{\cdot}10^{-2}$	$8.70{\cdot}10^{-2}$

σ_3 is small and approximately 90.7% of the total concentration of bound Hfq is bound to Qrr3.

To show that the results of the full model are consistent with those of the simplified model, the relative change in sensitivity in the wild-type strain from increasing a single parameter by 10% is measured. The parameter and corresponding relative change in sensitivity is shown in Figure 4.4. These results show that the relative change in the sensitivity in the full model is consistent with the analysis of the simplified model. For example, increasing V_{r_3} increases σ_3 in *V. cholerae* and increasing V_{r_2} decreases σ_2 in *V. harveyi*. The only exception is that perturbations in E_{q_3} increased, rather than decreased, σ_3 in *V. cholerae* and vise versa for E_{q_2} in *V. harveyi*. The results are similar for the other Qrr as well. Therefore, the mechanisms underlying Qrr redundancy suggested by the analysis of the simplified model are consistent with those underlying redundancy in the full model as well.

V. harveyi and *V. cholerae* Qrr are redundant when perturbations in Qrr are small and, in general, there are 30 different constraints that contribute to Qrr redundancy. To identify the mechanisms underlying Qrr redundancy in *V. harveyi* and *V. cholerae* when the perturbations in Qrr are small, the above computational analysis of the full model is extended. The relative change in σ_2 and σ_3 in *V. harveyi* and *V. cholerae*, respectively, is measured as the parameters are increased by 10%. The mechanisms that are responsible for redundancy are those that change the sensitivity by at least 10%. Overall, the model shows that σ_2 in *V. harveyi* and σ_3 in *V. cholerae* are most sensitive to changes in $V_{r_n}, \widehat{K}_{P_n}$, and E_{r_n}, as shown in Figure 4.5. The only exception is that *V. harveyi* σ_3 is sensitive to E_{q_3} as well. This shows that perturbations in the Qrr feedback (i.e., E_{o_n}, $\widehat{K}_{L_n}$, and V_{q_n}) do not significantly change the sensitivity. Therefore, the expression of *qrr* relative to *luxR/hapR* (V_{r_n}), affinity of LuxO-P to the *qrr* promoter ($\widehat{K}_{P_n}$), and the affinity of Qrr to *luxR/hapR* mRNA (E_{r_n}) underly Qrr redundancy in *V. harveyi* and *V. cholerae* when perturbations in Qrr are small.

Given that similar parameters underly redundancy in each species, the differences in the redundancy of Qrr between large vs. small perturbations in Qrr are, therefore, related to the saturation of Hfq, rather than to the differences in the mechanisms underlying each. Figure 4.3 shows that repression of LuxR, but not HapR, depends on the stoichiometry of Qrr at LCD. Furthermore, Table 4.1 shows that $S \approx 0$ and $S \approx 1$ for *V. harveyi* and *V. cholerae*, respectively, when all of the Qrr are present. The model also shows that Hfq remains mainly unsaturated in *V. harveyi* and saturated in *V. cholerae* at LCD when all

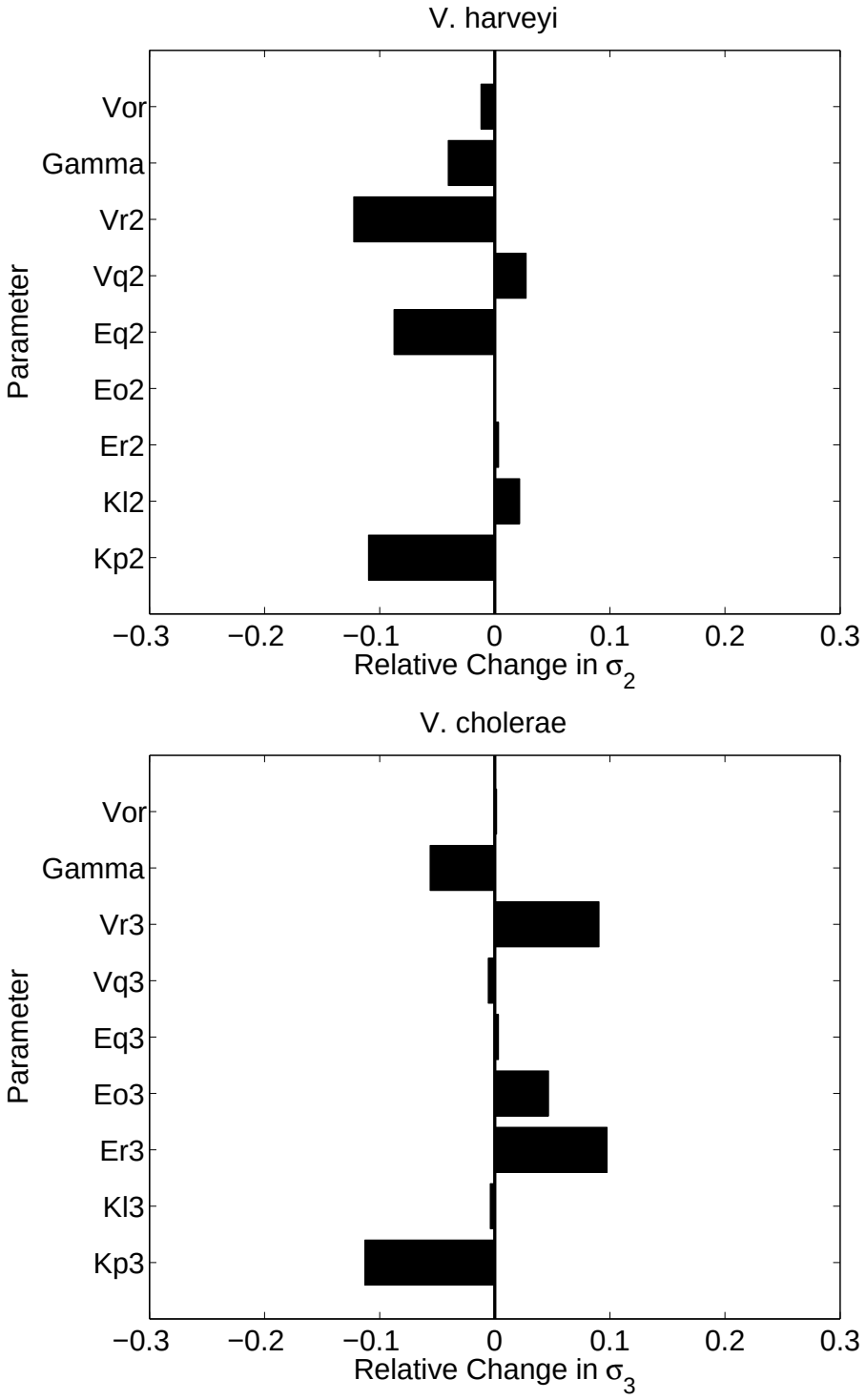

Figure 4.4. The relative change of σ_2 in *V. harveyi* and σ_3 in *V. cholerae* after increasing the indicated parameter by 10%. The change in sensitivity is relative to the sensitivity in the wild-type strain. The parameters are listed in Table 3.1 and $\Gamma = 10^5$.

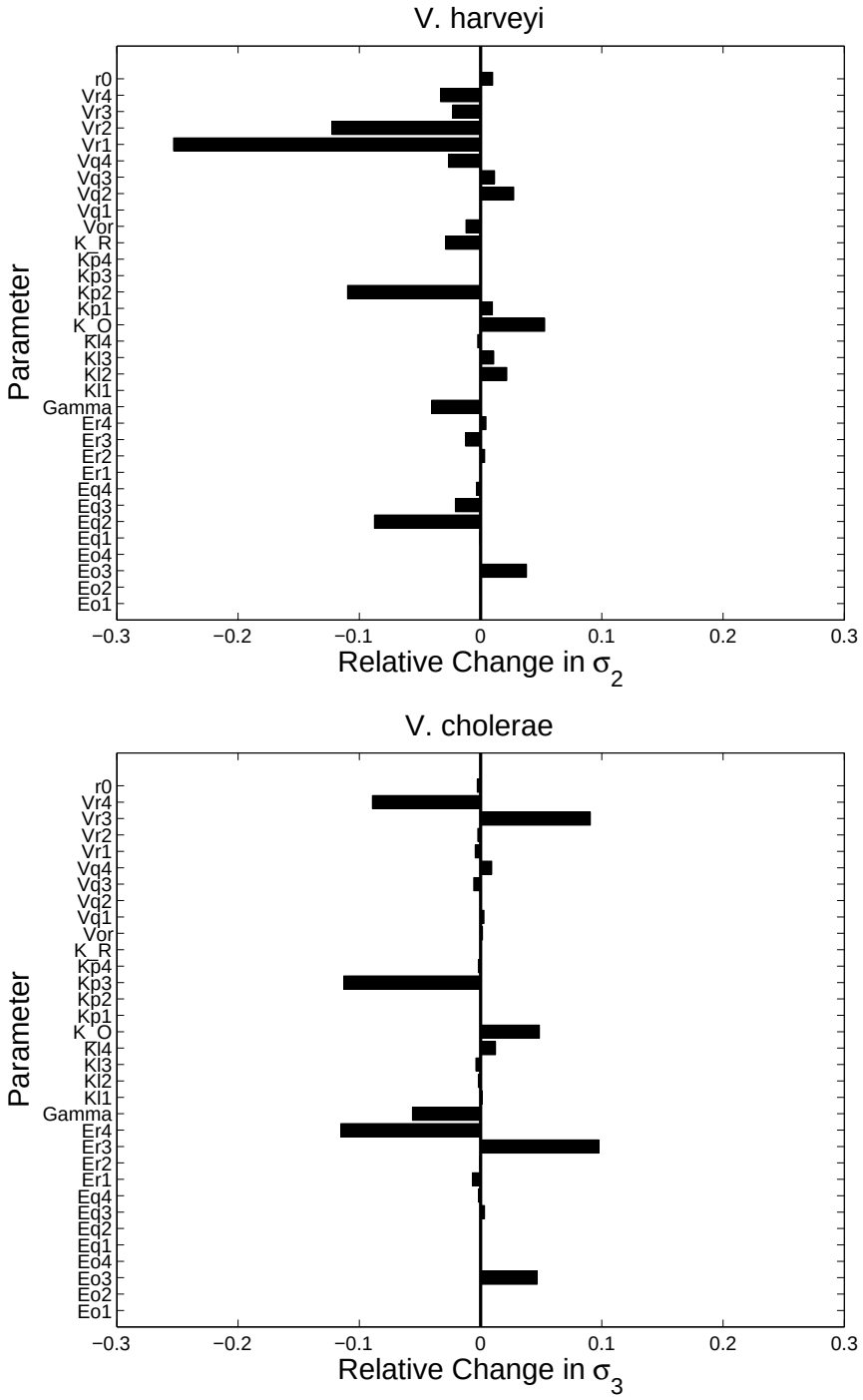

Figure 4.5. The relative change of σ_2 in *V. harveyi* and σ_3 in *V. cholerae* after increasing the indicated parameter by 10%. The change in sensitivity is relative to the sensitivity in the wild-type strain. The parameters that change σ by at least 10% are those that are associated with Qrr redundancy in *V. harveyi* and *V. cholerae*. The parameters are listed in Table 3.1 and $\Gamma = 10^5$.

but one Qrr are removed (not shown). This means that *V. cholerae* Qrr compete with one another at LCD to saturate Hfq. Additionally, the repression of HapR is relatively independent of the stoichiometry of Qrr because E_{r_n} are similar in *V. cholerae*. The Qrr in *V. harveyi*, however, neither saturate Hfq nor repress LuxR similarly, so *V. harveyi* Qrr are unable to compensate for large fluctuations in Qrr. Therefore, *V. cholerae* Qrr, rather than *V. harveyi* Qrr, are redundant to large perturbations in Qrr because *V. cholerae* Qrr saturate Hfq at LCD and repression of HapR is relatively independent of the stoichiometry of Qrr.

Although the same mechanisms underly redundancy in both species, they lead to different concentrations of Hfq-Qrr in *V. harveyi* and *V. cholerae*. For example, a 10% change in V_{r_n} corresponds to at least a 10% change in the sensitivity of LuxR/HapR to Qrr in *V. harveyi* and *V. cholerae*. In this sense, the expression of $luxR/hapR$ relative to qrr underlies Qrr redundancy in *V. harveyi* and *V. cholerae*. However, V_{r_n} is at least 10-fold less in *V. cholerae* than it is in *V. harveyi* and, hence, saturates Hfq with Qrr in *V. cholerae* more than in *V. harveyi*. Therefore, even though the relative expression of $luxR/hapR$ to qrr (V_{r_n}) underlies redundancy in both species, differences in V_{r_n} contribute to differences in the concentration of Hfq-Qrr between the species.

4.2.9 Qrr Redundancy is Independent of Dosage Compensation

Svenningsen et al. suggested that dosage compensation is the mechanism underlying Qrr redundancy [94]. However, analysis of the simplified model suggests that dosage compensation diminishes with sensitivity. This is also reflected in the full model.

The relative change in the *qrr* promoter activity (4.19) (i.e., dosage compensation) and the sensitivity are computed using the *V. cholerae* parameterization and $10^{-3} \leq \Gamma \leq 10^{5}$. The results, in Figure 4.6, show that the fold change in the *qrr* promoter activity decreases with the sensitivity. The results are qualitatively similar using the *V. harveyi* parameterization as well (not shown). Therefore, because dosage compensation is driven by changes in target mRNA levels and Qrr redundancy implies that the target mRNA levels are relatively constant, dosage compensation is not the mechanism underlying Qrr redundancy.

4.2.10 Testing the Qrr Redundancy Hypotheses

To test the Qrr redundancy hypothesis of [94] with the model, the same experiment from Figure 4.3 is repeated using two different *V. cholerae* strains. The first strain has

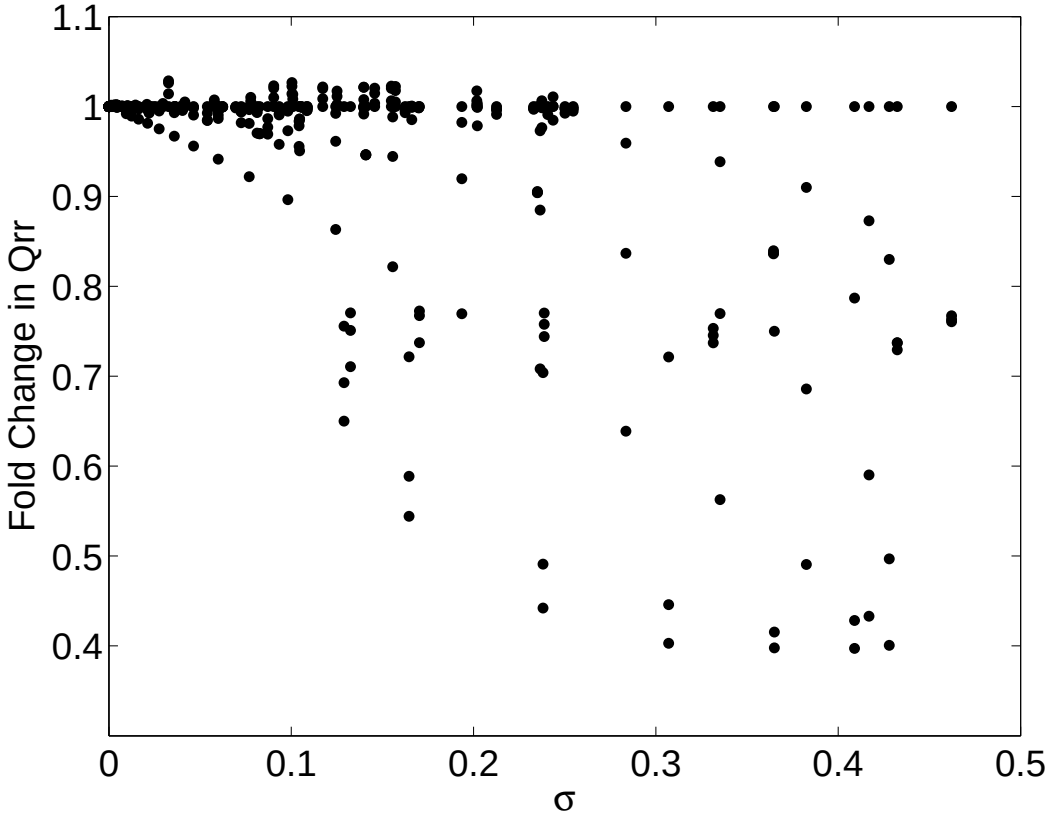

Figure 4.6. Relationship between the fold change in the *qrr* promoter activity and sensitivity in *V. cholerae* for $10^{-3} \leq \Gamma \leq 10^{5}$. Parameters are listed in Table 3.1. Qualitatively similar results were obtained using the *V. harveyi* parameters (not shown).

no Qrr feedback from setting $\widehat{K}_{L_n} = V_{q_n} = E_{o_n} = 0$. The second strain has an increased expression of *hapR* relative to *qrr* from increasing V_{r_n} 10-fold. If either the Qrr feedback or the expression of *hapR* relative to *qrr* is a mechanism underlying Qrr redundancy, then the aforementioned changes to the parameters will diminish *V. cholerae* Qrr redundancy.

Figure 4.7 (top) shows that removing the Qrr feedback tends to increase rather than decrease Qrr redundancy in *V. cholerae*. Conversely, Figure 4.7 (bottom) shows that Qrr redundancy is abolished when the expression of *hapR* relative to *qrr* is increased. Importantly, the *V. cholerae* response in this latter case is qualitatively similar to that of *V. harveyi* (compare Figure 4.3, top, with Figure 4.7, bottom). Therefore, the expression of *hapR* relative to *qrr*, rather than dosage compensation, is a mechanism underlying Qrr redundancy in *V. cholerae*.

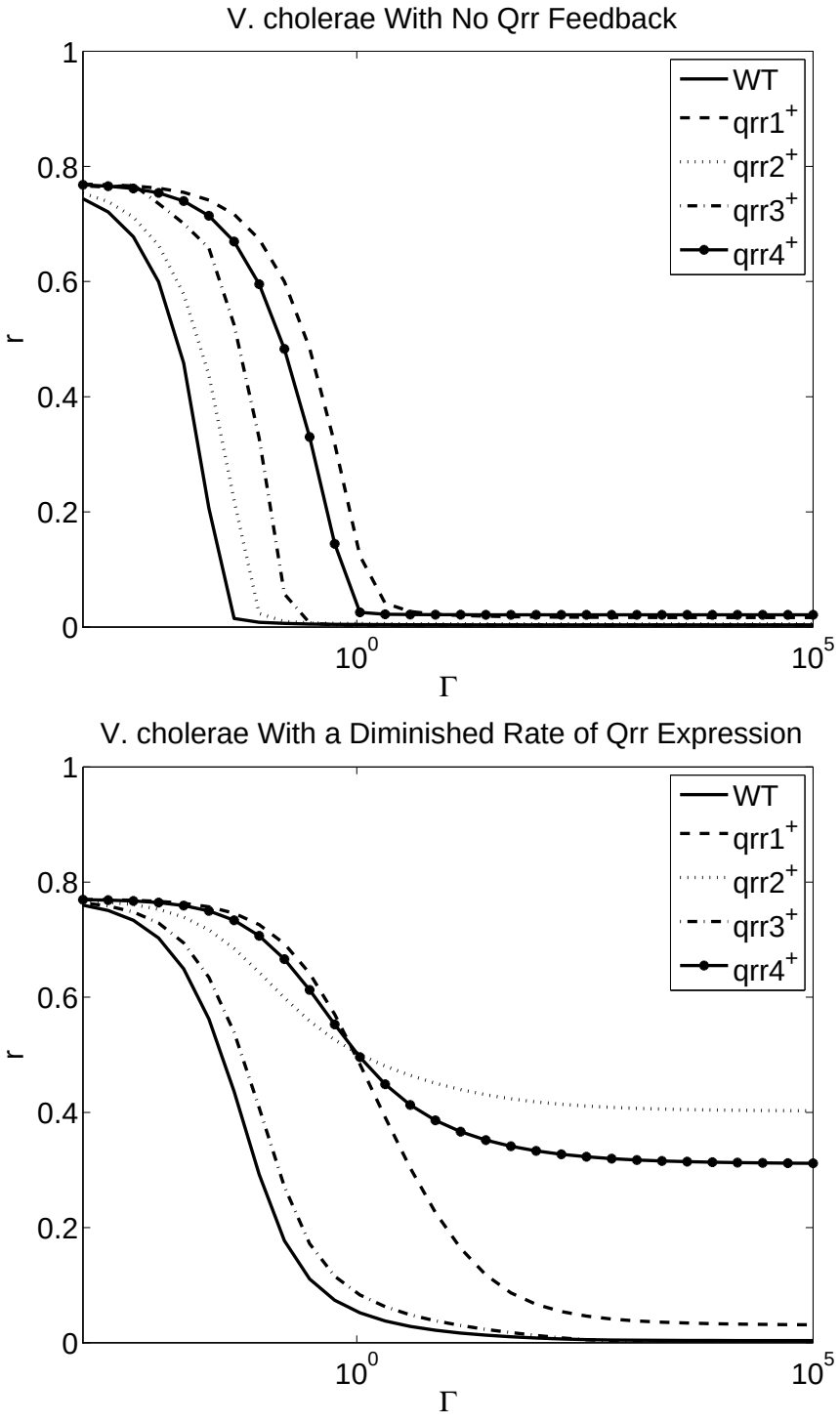

Figure 4.7. HapR expression in a *V. cholerae* wild-type strain and isogenic strains where all but one Qrr are removed. Increasing the expression of *hapR* relative to *qrr* (bottom), rather than removing the Qrr feedback (top), abolishes the redundancy phenotype in *V. cholerae*. Parameters are listed in Table 3.1.

4.3 Conclusion

The mechanisms underlying the additive and redundant Qrr phenotypes in *V. harveyi* and *V. cholerae* are unknown given the similarities between their quorum sensing systems. The current hypothesis in the literature proposes that the phenotypes arise from differences in the dosage compensation between *V. harveyi* and *V. cholerae* [94]. To our knowledge, this hypothesis remains untested, although it continues to be discussed in the current literature [8, 33, 85, 91]. In this work, the mechanisms underlying Qrr redundancy in *V. harveyi* and *V. cholerae* are identified and the dosage compensation hypothesis tested.

Qrr are redundant when the sensitivity of LuxR/HapR with respect to Qrr is small. This definition consistent with the interpretation of the additive and redundant Qrr phenotypes in the literature and is used to measure Qrr redundancy in the model. There are two different criteria leading to Qrr redundancy. In the first case, Qrr are redundant when there is a strong LuxO-Qrr feedback, weak LuxR/HapR-Qrr feedback, weak affinity of LuxO-P to the *qrr* promoter, high expression of *luxO* relative to *luxR/hapR*, and/or a high expression of *luxR/hapR* relative to *qrr*. These factors cause Qrr redundancy by limiting the concentration of Hfq-Qrr and, hence, the repression of LuxR/HapR by Qrr. In the second case, Qrr are redundant when there is a weak LuxO-Qrr feedback, strong LuxR/HapR-Qrr feedback, low expression of *luxO* relative to *luxR/hapR*, low expression of *luxR/hapR* relative to *qrr*, high binding affinity of *qrr* to Hfq, and/or when the binding affinities for each *qrr* to *luxR/hapR* are approximately equal. In this case, Qrr redundancy reflects that Hfq is saturated with Qrr and repression of LuxR/HapR is independent of the stoichiometry of the Qrr. Importantly, these results show that differences in Qrr feedback are neither necessary nor sufficient to explain Qrr redundancy and is in contrast to [94].

The behavior of the model independently produces qualitatively similar additive and redundant Qrr phenotypes for *V. harveyi* and *V. cholerae* to those in the literature. This result acts as validation of the model and strengthens its correspondence with the biology. Furthermore, this result also supports the hypothesis that the additive and redundant Qrr phenotypes originate from kinetic differences between the *V. harveyi* and *V. cholerae* sRNA circuits [57, 102, 94].

The model predicts that *V. harveyi* and *V. cholerae* Qrr are redundant at LCD when the changes in Qrr levels are small. Consequently, the affinity of Qrr to *luxR/hapR* mRNA and the expression of *qrr* relative to *luxR/hapR* lead to the redundancy of Qrr in both species. Therefore, the additive and redundant Qrr phenotypes emerge from differences in the concentrations of Hfq-Qrr between the species and dosage compensation diminishes as

the sensitivity of LuxR/HapR to Qrr diminishes in *V. harveyi* and *V. cholerae*. Increasing the expression of *hapR* relative to *qrr*, rather than removing the Qrr feedback, abolishes Qrr redundancy in *V. cholerae* and supports our alternate hypothesis.

To our knowledge, this is the first detailed model of the *V. harveyi* and *V. cholerae* sRNA circuit to identify the mechanisms underlying Qrr redundancy and to test the Qrr redundancy hypothesis of [94]. Contrary to their hypothesis, dosage compensation is neither necessary nor sufficient for Qrr redundancy in *V. harveyi* and *V. cholerae*. Rather, expression of *luxR*/*hapR* relative to *qrr* and the affinity of Qrr to *luxR*/*hapR* mRNA underly redundancy in both species. Furthermore, the additive and redundant Qrr phenotypes emerge from differences in the concentration of Hfq-Qrr. Although the concentration of Hfq is constant in the model, environmental factors might regulate its availability for quorum sensing [46]. Altogether, this suggests that Hfq has a significant role in quorum sensing (for an alternate reasoning see [33]). This study does not address other mechanisms that might underly the additive and redundant Qrr phenotypes as well such as environmental factors [84, 87, 55, 56], factors up/downstream of the sRNA circuit [33], or temporal differences between the *V. harveyi* and *V. cholerae* quorum sensing systems. Lastly, further experimentation is needed to verify the results.

CHAPTER 5

CONCLUSION

Bacteria were once thought to act as individuals; however, experiments over the last 40 years have shown that they can coordinate their behavior with other bacteria based on the local cell-population density. Most of the research in this field has focused on understanding the gene regulatory mechanisms underlying these quorum sensing systems, although recent research is investigating the ecological benefits and evolutionary stability of quorum sensing systems. There are different variations of quorum sensing systems in bacteria, but each is comprised of any number of only three canonical quorum sensing circuits. LuxIR-type circuits have a nonlinear positive feedback loop governing the synthesis of autoinducer. This feedback loop has been shown to facilitate a robust response, although experimental validation of this behavior has only been observed in a few bacteria. The remaining two canonical quorum sensing circuits, two-component-type and hybrid circuits, are functionally identical to one another and respond in a graded manner.

The hybrid quorum sensing circuits in *V. harveyi* and *V. cholerae* regulate expression of their respective virulence factors and are essential for the full efficacy of their virulence response. Their quorum sensing systems are comprised of a phosphorelay cascade that relays information about the local cell-population density to the sRNA circuit. The sRNA circuit regulates expression of the master transcriptional regulator that is responsible for regulating genes downstream of the quorum sensing system. The *V. cholerae* quorum sensing system is topologically identical to that of *V. harveyi* and the components of their quorum sensing systems are homologous. Although their quorum sensing systems are similar, *V. harveyi* and *V. cholerae* respond differently under identical experimental conditions. In particular, *V. harveyi* Qrr are additive because all of its Qrr are needed to repress LuxR. The *V. cholerae* Qrr, however, are redundant because any one of its Qrr is sufficient to repress HapR.

Experimentalists propose that the additive and redundant Qrr phenotypes arise from subtle tuning differences between their sRNA circuits. This hypothesis, however, is difficult,

time consuming, and costly to verify experimentally. In this work, a novel mathematical model of the *V. harveyi* and *V. cholerae* sRNA circuit was formulated then parameterized to identify parametric differences that underly the phenotypic differences. Chapter 3 details the parameterization of the model. Overall, the behavior of the model agrees quantitatively with a variety of empirical data from *V. harveyi* and *V. cholerae*. This suggests that the current understanding of the biology is sufficient to understand quorum sensing in *V. harveyi* and *V. cholerae*. Analysis of the linearized forward map shows that there are, respectively, four and eight weak search directions in the parameter estimation of *V. harveyi* and *V. cholerae*. This means that more experimentation is needed to complete the model. Since the parameter estimation is robust, different experiments with new information are needed to complete the model.

Parameter estimates are often needed to understand the behavior of a particular instance of a system and to know whether the results are biologically relevant. Most parameters are unknown, however, and is a common limitation cited in modeling papers of LuxIR-type circuits [3, 17, 16, 39, 40, 83], phosphorelay models [7, 65, 62], and sRNA models [28, 67, 58, 5, 66, 60, 69]. In particular, although LuxIR-type circuits are bistable under certain parametric constraints, this behavior has not been observed in *P. aeruginosa*, for example [88]. Most parameters in this work are reliably estimated and, as such, can aid in the development of quorum sensing and related models. Furthermore, the parameter estimation work flow and analysis of its robustness and weak search directions serve as an example for future work.

A related area of active research is to develop methods that reduce the uncertainty of the parameter estimates [14, 4, 6, 107]. These methods are based on a Bayesian (Monte-Carlo) analysis because this approach is also used to estimate parameters and their corresponding uncertainties. In general, however, this analysis is computationally intensive and, in the case of the sRNA model, impractical because the forward map is costly to evaluate. Therefore, the work in Chapter 3 suggests that the left-singular vectors associated with the smallest singular values can be used to identify the set of experiments where more information is needed. The reasoning behind this result is analogous to using the right-singular vectors associated with the smallest singular values to identify the weak search directions. Although the singular vectors are derived from the linearization of the forward map evaluated at a specific parameterization, this approach might, nevertheless, be an appropriate, intermediate analysis applicable in cases where a full Bayesian analysis is impractical.

Chapter 3 shows that the behavior of the model is quantitatively consistent with a variety

of empirical data. This means that the model can be used for *in silico* design, testing, and analysis of experiments. Hence, it can be used to identify experiments with new information to complete the model. For example, there is some debate concerning whether the sRNA within *V. harveyi* and *V. cholerae* repress target mRNA equally [102, 94, 44, 33]. The sRNA model be used to design and predict the outcome of an experiment testing this then the results can be validated in the lab. In this sense, the model can be viewed as a utility to identify experiments that can complete the model and formulate new, experimentally verifiable hypotheses. This application of the model is similar to that cited in the early models of LuxIR-type circuits [47, 24, 11].

Efforts to parameterize models that are more complex the sRNA model in this work are also underway. Recently, for example, researchers have developed a whole-cell model representing the expression of the 525 genes in *Mycoplasma genitalium* then used data from more than 900 scientific papers to parameterize the model [50] (for models of a similar scope see [29, 53, 18, 100]). These models are useful to understand the causes and treatments of emergent diseases such as cancer, parkinson's disease, and alzheimer disease.

The model was analyzed in Chapter 4 to understand the general mechanisms underlying the additive and redundant Qrr phenotypes as well as those that are specific to *V. harveyi* and *V. cholerae*. The results suggest that the additive and redundant Qrr phenotypes are an emergent phenomenon and, in the case of *V. harveyi* and *V. cholerae*, reflect differences in the saturation of Hfq with Qrr. Preliminary analysis of the model showed that the concentration of Hfq-Qrr is necessarily very large or very small when Qrr are redundant. Subsequent analysis showed that there were up to 30 different combinations of parametric constraints that could lead to Qrr redundancy, so the parametric constraints underlying Qrr redundancy depend on the particular species.

Although the model independently produces the additive and redundant Qrr phenotypes that are qualitatively similar to the data, the model suggests that *V. harveyi* and *V. cholerae* Qrr are redundant when the perturbations in Qrr are small. Further analysis showed that the parameters associated with the expression of *qrr* relative to *luxR/hapR*, affinity of *qrr* to *luxR/hapR* mRNA, and the affinity of LuxO-P to the *qrr* promoter were responsible for Qrr redundancy in both species. However, differences in the *V. harveyi* and *V. cholerae* estimates of these parameters lead to different total concentrations of Hfq-Qrr. Therefore, the additive and redundant Qrr phenotypes reflect differences in the total concentration of Hfq-Qrr between the two species.

Even though Hfq and Qrr feedback have been implicated in the mechanism(s) underlying

the additive and redundant Qrr phenotypes [94, 33], this work presents the first analysis of a mathematical model that incorporates either of these features. Fenley et al. suggested that differences in the availability of Hfq could be responsible for the additive and redundant Qrr phenotypes. In their simplified model of the sRNA circuit, they assumed that the sRNA circuits in *V. harveyi* and *V. cholerae* are identical (implying that there are no parametric differences) and ignored both Hfq and Qrr feedback. To produce the additive and redundant phenotypes with their model, they suggested that environmental factors acted to establish different activation thresholds for the expression of LuxR/HapR by Hfq limiting the repression of target mRNA or other factors [33, 44]. Hence, their argument implicating Hfq is indirect and necessitated by the assumption that there are no parametric differences between *V. harveyi* and *V. cholerae*. By contrast, in this work Hfq was first implicated following from the steady–state analysis of the model and the parameter estimates.

Svenningsen et al. showed that Qrr feedback is responsible for increasing Qrr expression when one or more Qrr are removed and argued that a strong Qrr feedback is the mechanism underlying Qrr redundancy [94]. Contrary to their hypothesis, this work shows that Qrr can be redundant when the LuxR/HapR-Qrr feedback is stronger than the LuxO-Qrr feedback, or vise versa, rather than some synergy between the two. Furthermore, this work shows that Qrr feedback is neither necessary nor sufficient for Qrr redundancy. Most importantly, however, the model shows that the more redundant the Qrr, the smaller the dosage compensation. This counter-intuitive result is explained by the fact that dosage compensation reflects a change in target mRNA levels, yet target mRNA levels are relatively constant when Qrr are redundant. Other than fine-tuning the rate and onset of the LCD to HCD transition [103, 94, 57, 102, 104], the role of the Qrr feedback remains unknown.

This work and other investigations have not addressed the ecological benefits and evolutionary stability of the additive/redundant Qrr phenotype. For example, although Qrr feedback is not important for Qrr redundancy, the Qrr feedback might act to dampen noise during the LCD to HCD transition. The additivity of *V. harveyi* Qrr might explain why it targets hosts with weakened immune systems. Similarly, the redundancy of *V. cholerae* Qrr might act to facilitate a robust transition to/from LCD and HCD mode and aid in its rapid dissemination. These and other open questions can be investigated using the sRNA model in this work.

Lastly, this work contributes to the broader understanding of the role and function of noncoding regulatory RNA. Less than 5% of the human genome codes for proteins, which is only 2- to 3-fold more than some species of bacteria [89], yet approximately 66% of the

human genome is transcribed into RNA [54]. The role of these noncoding RNA remains an active area of research, for they are thought to be responsible for the higher complexity of humans [89]. The discovery of noncoding regulatory RNA in simple organisms along with bioinformatics tools helps to discover noncoding RNA in complex organisms and their function [54]. Some noncoding RNA, such as Qrr in *V. harveyi* and *V. cholerae*, are known to regulate gene expression. This work argues that Hfq modulates Qrr-facilitated repression of mRNA and is essential to create a robust redundant Qrr phenotype.

APPENDIX A

MATLAB CODE: *V. harveyi* PARAMETERIZATION

The following is a collection of the essential Matlab code used to parameterize *V. harveyi*.

A.1 Main *V. harveyi* Parameterization Code

```
function varargout = paramVHARVEYI(varargin)
% Run this to parameterize the model to the V. harveyi data

% Files representing the V. harveyi raw data
% Tu2010LuxRvsAIAllQrr_RawData.mat
% Tu2010LuxRvsAIQrr4Only_RawData.mat
% TuLuxRqrrFeedbackProgressiveKnockout_RawData.mat

%% Reset the seed for the random number generator based on
%% the time of day.
RandStream.setDefaultStream(RandStream('mt19937ar','seed',...
            sum(100*clock)));
global fileNameAppend
global computeAbsoluteError makeErrAMatrix
global globalTolerance

computeAbsoluteError = false;
makeErrAMatrix = false;

fileNameAppend = 'a_filename_';
```

```
% % Generate the data sets for the robust analysis
% solnDistribution

theNoise = 0;
[allData,p]  = initializeVharveyi('add noise',theNoise);
states = [];
%% Run this function to parameterize the model to the V. harveyi data
masterMainIterates(p,states,allData);

%=========================================================================
function masterMainIterates(p,states,allData)
global fileNameAppend
global globalTolerance

thePlan = plans('VhOpt','numGam',2);
numParams = getNumVarsInPlan(thePlan);

%% General optimization parameters and preferences
maxIter = 25; % # of iterations/lsqnonlin iterate
maxEvals = 1000000;
tol = 1e-14;
genTol = 1e-15;
optimality = tol;
dispStr = 'iter';
ctrMax = 1; % # of repetitions of lsqnonlin
globalTolerance = tol;

numEst = 10;
numSolns = 15;

diaryFileName = @(n)[fileNameAppend 'DiaryForEstimate' num2str(n) '.txt'];
fileName = @(n) [fileNameAppend 'mainVhSoln_JacobianIterates_' ...
                            num2str(n) '.mat'];
```

```
for i = 1:numSolns
    diary(diaryFileName(i))
    tic
    disp(['Current solution attempt #' num2str(i)])

    files = dir('VHguessRanges.txt');
    % Get the date for the most recent initial guess domain
    soln.guessFileDate = files.date;

    [soln.bestEst, soln.bestNorm, soln.estData] = ...
          findBestMainEstimateIterates(allData,p,numEst,states,numParams);

    %% Parameterize the model to the V.harveyi data
    [soln.X2, soln.err, soln.opt] = generalOptRoutine(
        @(x)masterIterateOptimization(x,states,p,allData),...
        soln.bestEst,'maxIter',maxIter,'maxEvals',maxEvals,...
        'normTol',tol,'genTol',genTol,'dispStr',dispStr,...
        'optimality',optimality,'ctrMax',ctrMax,'computeJacobian',true);

    %% Compute the norm of the error
    soln.normSoln = norm(soln.err);

    %% Save the result
    cd('C:\..\MATLAB\VH Data')
    save(fileName(i),'soln')
    cd('C:\..\MATLAB\VHarveyiParamCode')

    theTime = toc/60;
    disp(['Running time: ' num2str(theTime) ' min'])
    diary off
end

%=========================================================================
function solnDistribution
```

```
%% Find the variance of the V. harveyi model parameters to determine
%% the robustness of the parameters
global fileNameAppend
global globalTolerance

noiseRange = [0.1];
thePlan = plans('VhOpt','numGam',2);

maxIter = 10; % # of iterations/lsqnonlin iterate
maxEvals = 1000000;
tol = 1e-14;
genTol = 1e-15;
optimality = tol;
dispStr = 'iter';
ctrMax = 1; % # of repetitions of the lsqnonlin
numNoiseSamples = 48;
globalTolerance = tol;

diaryFileName = @(n)[fileNameAppend 'SampleDiary' num2str(n) '.txt'];
fileName = @(n,m) [fileNameAppend 'VhD_' num2str(n) '_' num2str(m)'.mat'];

pWT = makeStrain([],{'WT V.harveyi'});
pInit = stovec(pWT,thePlan);

% A function used to randomly perturb the data by at most 10%
noiseAmt = 0.1;
addNoise = @(theData)( theData.*(1+noiseAmt*(2*rand(size(theData))-1) ) );

for j = 1:length(noiseRange)
    noiseAmt = noiseRange(j);
    for i = 1:numNoiseSamples
        [allData,p,states]  = initializeVharveyi;

    %% Randomly perturb data.
```

```
    allData.dataStep1 = addNoise(allData.dataStep1);
    allData.dataStep2 = addNoise(allData.dataStep2);
    allData.Tu2008LuxRFeedback.Y = addNoise(allData.Tu2008LuxRFeedback.Y);
    allData.Tu2010AllQrr4Only.Y = addNoise(allData.Tu2010AllQrr4Only.Y);
    allData.Tu2010AllQrr.Y = addNoise(allData.Tu2010AllQrr.Y);

        tic
        disp(['Current sample #' num2str(i)])
        soln.initEst = pInit;
        initEstResult = masterIterateOptimization(...
                                  pInit,states,pWT,allData);
        soln.initEstNorm = norm(initEstResult(:));

        disp('Starting the parameterization')
        [soln.X2, soln.err, soln.opt] = generalOptRoutine(
            @(x)masterIterateOptimization(x,states,pWT,allData),...
            pInit,'maxIter',maxIter,'maxEvals',maxEvals,'normTol',tol,...
            'genTol',genTol,'dispStr',dispStr,'optimality',optimality,...
            'ctrMax',ctrMax,'computeJacobian',true);

        soln.normSoln = norm(soln.err(:));
        soln.noiseAmt = noiseAmt;

        cd('C:\..\MATLAB\VH Data\Vh Distributions')
        save(fileName(i,j),'soln')
        cd('C:\..\MATLAB\VHarveyiParamCode')
        runTime = toc;
        disp(['Running Time: ' num2str(runTime/60) ' min'])
    end
end

%=========================================================================
function [bestEst, bestNorm, estData] = ...
        findBestMainEstimateIterates(allData,p,numEst,states,numParams)
```

```
sampleResult = masterIterateOptimization(...
                rand(numParams,1),states,p,allData);
numData = length(makeVect(sampleResult));
estData.x = zeros(numParams,numEst);
estData.err = zeros(numData,numEst);
estData.norm = zeros(1,numEst);
i = 1;
while i <= numEst
    disp(['Working on main estimate #' num2str(i)])
        estData.norm(i) = norm(estData.err(:,i));
        i = i+1;
end
[theMin, at] = min(estData.norm);
bestEst = estData.x(:,at);
bestNorm = estData.norm(at);
disp(['Norm of the best estimate is: ' num2str(bestNorm)])

%=========================================================================
function [iterateError, Derror] = ...
            masterIterateOptimization(Z0,states,p,allData)
p = updateIterateGuess(Z0,states,p,allData);
[iterateError,Derror] = harveyiSingleIterate(p,states,allData);

%=========================================================================
function [p,states] = updateIterateGuess(Z0,states,p,allData)
thePlan = plans('VhOpt','numGam',2);
p = vectos(Z0,thePlan);
```

A.2 Initialize the *V. harveyi* Optimization

```
function [allData,p,varargout]  = initializeVharveyi(varargin)
% Returns all of the V. harveyi data and objects for its optimization.

% Known parameters found in previous articles
```

```
knownParams = getKnownParams;
numsRNA = 4;

for i = 1:2:length(varargin)
    switch varargin{i}
        case lower('add noise')
            noiseAmt = varargin{i+1};
        otherwise
            error('Unknown option in initializeVharveyi')
    end
end

load('Tu2010LuxRvsAIAllQrr_RawData.mat')
for i = 1:4
    Tu2010AllQrr.Y(i,:) = Data(1,i).Y;
    Tu2010AllQrr.X(i,:) = Data(1,i).X;
end
Tu2010AllQrr.Y = VHformatAllQrrData(Tu2010AllQrr.Y);

load('Tu2010LuxRvsAIQrr4Only_RawData.mat')
for i = 1:4
    Tu2010AllQrr4Only.Y(i,:) = Data(1,i).Y;
    Tu2010AllQrr4Only.X(i,:) = Data(1,i).X;
end
Tu2010AllQrr4Only.Y = VHformatAllQrrData(Tu2010AllQrr4Only.Y);

load('TuLuxRqrrFeedbackProgressiveKnockout_RawData.mat')
for i = 1:5
    Tu2008LuxRFeedback.Y(i,:) = Data(1,i).Y;
    Tu2008LuxRFeedback.X(i,:) = Data(1,i).X;
end
Tu2008LuxRFeedback.Y = VHformatLuxRQrrFeedbackData(Tu2008LuxRFeedback.Y);

% Get the data for both steps
```

```
dataStep1 = VHgetStep1Data(numsRNA);
dataStep2 = VHgetStep2Data(numsRNA);

% Get the params
p = getParams('new');
p = getParams('updateParams','numsRNA',numsRNA);

% Weights for each experiment
allData.knownParams = knownParams;
allData.Tu2010AllQrr = Tu2010AllQrr;
allData.Tu2010AllQrr4Only = Tu2010AllQrr4Only;
allData.dataStep1 = dataStep1;
allData.dataStep2 = dataStep2;
allData.numsRNA = numsRNA;
allData.Tu2008LuxRFeedback = Tu2008LuxRFeedback;
allData.residualKey = residualKey('species','Vh');
```

A.3 Model of All *V. harveyi* Experiments

```
function [x, dx] = harveyiSingleIterate(p,states,allData)
% This function represents F(p) and DF(p) for V. harveyi
%
% pWT: Structure containing the WT parameters
% states: Structure containing estimates of the steady-states for given
% parameterizations
% allData: Structure containing all of the V. harveyi data

if nargout ==1
    step2Error = VHtoOptStep2(p,'allData',allData,...
                        'expt data',allData.dataStep2);

    AI = allData.Tu2010AllQrr.X(1,:);
    exptData = allData.Tu2010AllQrr.Y;
    ptr = 4;
    step3Error = luxRvsAI(p,AI,'allData',allData,'expt data',...
```

```
                        exptData,'state pointer',ptr);

    AI = allData.Tu2010AllQrr4Only.X(1,:);
    exptData = allData.Tu2010AllQrr4Only.Y;
    ptr = 8;
    pMut = makeStrain(p,{'qrri RNA'},{1:3});
    step4Error = luxRvsAI(pMut,AI,'allData',allData,'expt data',...
                        exptData,'state pointer',ptr);
else
    [step2Error, Dstep2] = VHtoOptStep2(p,'allData',allData,...
                        'expt data',allData.dataStep2);

    AI = allData.Tu2010AllQrr.X(1,:);
    exptData = allData.Tu2010AllQrr.Y;
    ptr = 4;
    [step3Error, Dstep3] = luxRvsAI(p,AI,'allData',allData,'expt data',...
                        exptData,'state pointer',ptr);

    AI = allData.Tu2010AllQrr4Only.X(1,:);
    exptData = allData.Tu2010AllQrr4Only.Y;
    ptr = 8;
    pMut = makeStrain(p,{'qrri RNA'},{1:3});
    [step4Error, Dstep4] = luxRvsAI(pMut,AI,'allData',allData,...
                        'expt data',exptData,'state pointer',ptr);

    % Jacobian when the steady states are given
    dx = [Dstep2*allData.residualKey.step2Error.weight;
        Dstep3*allData.residualKey.step3Error.weight;
        Dstep4*allData.residualKey.step4Error.weight];
end
x = [step2Error(:)*allData.residualKey.step2Error.weight;
    step3Error(:)*allData.residualKey.step3Error.weight;
    step4Error(:)*allData.residualKey.step4Error.weight];
```

A.4 Model of the Experiment in Figures 3.3 and 3.4

```
function [f, df, varargout] = luxRvsAI(p,AI,varargin)
% Plot LuxR expression vs. AI concentration.
% Models the experiment in Tu et al. 2010
global computeAbsoluteError

statePtr = 0;
if nargin >=3
    for i = 1:2:length(varargin)
        switch lower(varargin{i})
            case lower('expt data')
                exptData = varargin{i+1};
            case lower('sdy states')
                states = varargin{i+1};
            case lower('state pointer')
                statePtr = varargin{i+1};
                if statePtr ~= 4 && statePtr ~= 8
    error(['state pointer needs to be either 4 or 8: ' num2str(statePtr)])
                end
            case lower('allData')
                allData = varargin{i+1};
            otherwise
                error('Wrong string argument in luxRvsAI')
        end
    end
end

if statePtr == 0
 error('Need to define statePtr to calculate the weights for the errors.')
end

if ~exist('allData','var') && exist('states','var') && ...
                exist('statePtr','var')
    error('Missing allData, states, and statePtr')
```

```
end
p.AI = AI;
pWT = p;
pNoLuxOAuto = makeStrain(pWT,{'LuxO Auto'});
pNoLuxOQrr = makeStrain(pWT,{'LuxO-qrr feedback'});
pNoLuxORegulation = makeStrain(pWT,{'LuxO Auto','LuxO-qrr feedback'});

if exist('states','var') % You're given the steady states
    theNames = fieldnames(states);
    statesWT = states.(theNames{statePtr});
    statesOAuto = states.(theNames{statePtr+1});
    statesOFeed = states.(theNames{statePtr+2});
    statesMut = states.(theNames{statePtr+3});

    ssWT = vectToSS(statesWT);
    ssOAuto = vectToSS(statesOAuto);
    ssOFeed = vectToSS(statesOFeed);
    ssOMut = vectToSS(statesMut);

    if nargout == 3
        t=0;
        error('This uses "hybrid", which is outdated...')
        errorWTAISS = makeVect(hybrid(t,statesWT,pWT,pWT.Gamma))...
            *allData.residualKey.(theNames{statePtr}).weight;

        % 3b) No LuxO Autoregulation
        errorNoLuxOAutoSS = makeVect(hybrid(t,statesOAuto,pNoLuxOAuto,...
            pNoLuxOAuto.Gamma))*allData.residualKey.(...
                    theNames{statePtr+1}).weight;

        % 4) No LuxO-Qrr feedback
        errorNoLuxOQrrSS = makeVect(hybrid(t,statesOFeed,pNoLuxOQrr,...
            pNoLuxOQrr.Gamma))*allData.residualKey.(...
                    theNames{statePtr+2}).weight;
```

```
        % 5) No LuxO Regulation
        errorNoLuxORegulationSS = makeVect(hybrid(t,statesMut,...
            pNoLuxORegulation,pNoLuxORegulation.Gamma))...
                    *allData.residualKey.(theNames{statePtr+3}).weight;

        varargout{1} = [errorWTAISS;errorNoLuxOAutoSS;...
                        errorNoLuxOQrrSS;errorNoLuxORegulationSS];
    end
else % otherwise compute the steady states
    % WT expression
    ssWT = getSSAt(pWT);

    % No LuxO Autoregulation
    ssOAuto = getSSAt( pNoLuxOAuto );

    % No LuxO-Qrr Feedback
    ssOFeed = getSSAt( pNoLuxOQrr );

    % No LuxO-Qrr Feedback
    ssOMut = getSSAt( pNoLuxORegulation );
end

RExpression = modelExpt(ssWT, ssOAuto, ssOFeed, ssOMut);
if nargout >=2
    df = computeJack(pWT ,ssWT, pNoLuxOAuto,ssOAuto, pNoLuxOQrr , ...
                ssOFeed,pNoLuxORegulation, ssOMut);
    if exist('exptData','var')
        if ~computeAbsoluteError
            df = scaleJacobianByData(df,exptData);
        end
    end
end
if exist('exptData','var')
```

```
    f = computeError(RExpression,exptData);
else
    f = RExpression;
end

%=========================================================================
function f = modelExpt(ssWT, ssOAuto, ssOFeed, ssOMut)
f = VHformatAllQrrData([ssWT.r; ssOAuto.r; ssOFeed.r; ssOMut.r]);

%=========================================================================
function DF = computeJack(pWT ,ssWT, pNoLuxOAuto,ssOAuto,pNoLuxOQrr, ...
                                    ssOFeed,pNoLuxORegulation, ssOMut)
% Compute the jacobian associated with this experiment
planName = 'VhOptStep2Jack';
thePlan = plans(planName,'AI',pWT.AI,'numGam',2);
sigmaWT = sensitivityStatesToParams(ssWT,pWT,thePlan);
sigmaOAuto = sensitivityStatesToParams(ssOAuto,pNoLuxOAuto,thePlan);
sigmaOFeed = sensitivityStatesToParams(ssOFeed,pNoLuxOQrr,thePlan);
sigmaOMut = sensitivityStatesToParams(ssOMut,pNoLuxORegulation,thePlan);

dWT = diffModel(ssWT,sigmaWT,ssWT,sigmaWT);
dOAuto = diffModel(ssOAuto,sigmaOAuto,ssWT,sigmaWT);
dOFeed = diffModel(ssOFeed,sigmaOFeed,ssWT,sigmaWT);
dOMut = diffModel(ssOMut,sigmaOMut,ssWT,sigmaWT);

[numDensity,numParams] = size(dWT);
DF = zeros(4*numDensity,numParams);
DFctr = 1;
for i = 1:numDensity
    DF(DFctr,:) = dWT(i,:);
    DFctr = DFctr +1;

    DF(DFctr,:) = dOAuto(i,:);
    DFctr = DFctr +1;
```

```
    DF(DFctr,:) = dOFeed(i,:);
    DFctr = DFctr +1;

    DF(DFctr,:) = dOMut(i,:);
    DFctr = DFctr +1;
end

%=========================================================================
function df = diffModel(ss,DssDp,ssWT,DssWTDp)
% Derivative of the model data with respect to all of the parameters
rWTHCD = ssWT.r(end);
DrWTDpHCD = DssWTDp{end}(1,:);

numDensity = length(ssWT.r);
numParams = length(DrWTDpHCD);

df = zeros(numDensity,numParams);

for densityCtr = 1:numDensity
    df(densityCtr,:) = (rWTHCD*DssDp{densityCtr}(1,:)-ss.r(densityCtr)*...
                        DrWTDpHCD)/rWTHCD^2;
end
```

A.5 Model of the Experiment in Figure 3.2

```
function [f,df, varargout] = VHtoOptStep2(p,varargin)
% Model the experiment in Tu et al. 2008
global computeAbsoluteError

pWT = p;
pNoRQFeedback = makeStrain(p,{'hapR-Qrr Feedback'});
pNoRRepression = makeStrain(p,{'hapR mRNA'});

if nargin >=2
```

```
    for i = 1:2:length(varargin)-1
        switch lower(varargin{i})
            case lower('allData')
                allData = varargin{i+1};
            case lower('expt data')
                data = varargin{i+1};
            case lower('sdy states')
                states = varargin{i+1};
            otherwise
                error('Wrong string argument in VHtoOptStep2')
        end
    end
end

% Either get or compute the steady-states
if exist('states','var') % if you're given the "states" structure
    error('This code is obsolete. Update to reflect the new jacobian')
    ssWT = vectToSS(states.WT);
    ssNoRQFeedback = vectToSS(states.NoRQFeedback);
    ssNoRRepression = vectToSS(states.NoRRepression);

    if nargout == 3
        t=0;
        error('This uses "hybrid", which is outdated.')
        errorWTSS = makeVect(hybrid(t,states.WT,pWT,pWT.Gamma))...
            *allData.residualKey.WT.weight;
        errorNoRQFeedbackSS = makeVect(hybrid(t,states.NoRQFeedback,...
            pNoRQFeedback,pNoRQFeedback.Gamma))...
            *allData.residualKey.NoRQFeedback.weight;
        errorNoRRepressionSS = makeVect(hybrid(t,states.NoRRepression,...
            pNoRRepression,pNoRRepression.Gamma))...
            *allData.residualKey.NoRRepression.weight;
        ssError = [errorWTSS;errorNoRQFeedbackSS;errorNoRRepressionSS];
        varargout{1} = ssError;
```

```
    end

else % Otherwise, compute the steady-states with the "getSSAt" function
    ssWT = getSSAt(pWT);
    ssNoRQFeedback = getSSAt(pNoRQFeedback);
    ssNoRRepression = getSSAt(pNoRRepression);
end

f = step2ModelResults(ssWT,ssNoRQFeedback,ssNoRRepression);
if nargout >=2
    df = computeJack(pWT,ssWT,pNoRQFeedback,ssNoRQFeedback,....
                    pNoRRepression,ssNoRRepression);

    if exist('allData','var') && exist('data','var')
        if ~computeAbsoluteError
            df = scaleJacobianByData(df,data);
        end
    end
end

if exist('data','var')
    f = computeError(f,data);
end

%=========================================================================
function X = step2ModelResults(ssWT,ssNoRQFeedback,ssNoRRepression)
% Simulates the model to reproduce Figure 4 in Tu et al. (2008)
ss = ssWT;
relTo = ss.q(:,1); % Qrr WT value at LCD
X = [ss.q(:,1)./relTo ss.q(:,2)./relTo]; % a vector of ones

% No LuxR-Qrr Feedback
ss = ssNoRQFeedback;
Y = [ss.q(:,1)./relTo ss.q(:,2)./relTo];
```

```
% No degradation of LuxR by Qrr
ss = ssNoRRepression;
X = [X ss.q(:,1)./relTo ss.q(:,2)./relTo];
X = VHformatStep2Data([X Y]);

%=========================================================================
function DF = computeJack(pWT,ssWT,pNoRQFeedback,ssNoRQFeedback,...
                          pNoRRepression,ssNoRRepression)

% Compute the jacobian associated with this experiment
planName = 'VhOptStep2Jack';
thePlan = plans(planName,'numGam',2);

sigmaWT = sensitivityStatesToParams(ssWT,pWT,thePlan);
sigmaNoRQFeedback = sensitivityStatesToParams(ssNoRQFeedback,...
                          pNoRQFeedback,thePlan);
sigmaNoRRepression = sensitivityStatesToParams(ssNoRRepression,...
                          pNoRRepression,thePlan);

dWT = diffModel(ssWT,sigmaWT,ssWT,sigmaWT);
dNoRQFeed = diffModel(ssNoRQFeedback,sigmaNoRQFeedback,ssWT,sigmaWT);
dNoRRepression = diffModel(ssNoRRepression,sigmaNoRRepression,...
                          ssWT,sigmaWT);

[r,n] = size(dWT);
DF = zeros(3*r,n);
for i = 1:2
    k = 1 + r/2*(i-1);

    j = 1 + 3*r/2*(i-1);
    DF(j:j+3,:) = dWT(k:k+3,:);

    j = j+4;
```

```
    DF(j:j+3,:) = dNoRRepression(k:k+3,:);

    j = j+4;
    DF(j:j+3,:) = dNoRQFeed(k:k+3,:);
end

%=========================================================================
function df = diffModel(ss,DssDp,ssWT,DssWTDp)
%                \/---- "1" indicates LCD
dWTLCD = DssWTDp{1}(3:6,:);
relTo = ssWT.q(:,1); % WT LCD levels

numDensity = length(ssWT.r);

df = zeros(length(relTo)*numDensity,length(dWTLCD(1,:)));
ptr = 1;

for densityCtr = 1:numDensity
    for i = 1:length(relTo)
        df(ptr,:) = (relTo(i)*DssDp{densityCtr}(2+i,:)-...
                    ss.q(i,densityCtr)*dWTLCD(i,:))/relTo(i)^2;
        ptr = ptr +1;
    end
end

function Y = VHformatStep2Data(X)
% To format identically the raw data and the model results.
[r,c] = size(X);
Y = zeros(r,c);
Y(:,1:c/2) = X(:,1:2:end);
Y(:,c/2+1:end) = X(:,2:2:end);
```

APPENDIX B

MATLAB CODE: *V. cholerae* PARAMETERIZATION

The following is a collection of the essential Matlab code used to parameterize *V. cholerae*.

B.1 Main *V. cholerae* Parameterization Code

```
function varargout = FINALVcParams(varargin)
cd('C:\..\MATLAB\VC')

% Randomize the seed for the random number generator
RandStream.setDefaultStream(RandStream('mt19937ar',...
                    'seed',sum(100*clock)));

global fileNameAppend
global computeAbsoluteError makeErrAMatrix
global globalTolerance

computeAbsoluteError = false;
makeErrAMatrix = false;
fileNameAppend = 'a_filename_';
doOptimization = false;
doSingleIterate = false;

diaryFileName = @(n)[fileNameAppend 'DiaryForEstimate' num2str(n) '.txt'];


%%% LOAD DATA & WEIGHTS %%%
[allData,p]= VCinitialize;
```

```
if doSingleIterate
    thePlan = plans('VcholeraeOpt','numGam',1);
    p = makeStrain([],{'WT V.cholerae'});
    printParams(p)
    X = stovec(p,thePlan);
    [f,df] = mainToOpt(X,allData);

    if nargout >=1
        varargout{1} = f;
    end
    if nargout >=2
        varargout{2} = df;
    end
end

if doOptimization
    ansFileName = @(n)[fileNameAppend 'VCfullSoln' num2str(n) '.mat'];
    maxSamples = 1; %5
    numEstimates = 1; %25
    hold on
    for i = 1:numEstimates
        diary(diaryFileName(i))
        tic
        disp(['Current solution attempt #' num2str(i)])

        files = dir('VC*Ranges.txt');
        theSoln.guessFileDate = files.date;

        [bestNorm, bestEst, otherEst] = findBestEst(maxSamples,allData);
        disp('Best estimate is found')
        theSoln.otherEst = otherEst;
        theSoln.X0 = bestEst;
```

```
        theSoln.estNorm = bestNorm;

        [p, ERR, X, opt] = main(bestEst,allData);

        theSoln.X = X;
        theSoln.ERR = ERR;
        theSoln.ansNorm = norm(ERR);
        theSoln.opt = opt;
        theSoln.computeAbsoluteError = computeAbsoluteError;
        theSoln.allData = allData;

        cd('C:\..\MATLAB\VC Data')
        save(ansFileName(i),'theSoln')
        cd('C:\..\MATLAB\VC')

        theTime = toc/60;
        disp(['Running time: ' num2str(theTime) ' min'])
        diary off
    end
end

    %%%% Look at the stats for the distribution of Vc solution
%     load('VCsolution')
%     pStar = p;
%     numSamples = 1;
%     noiseAmt = 0.1;
%
%     ansFileName = @(n)[fileNameAppend 'VCnoiseSoln' num2str(n) '.mat'];
%     hold on
%     for i = 1:50
%         diary(diaryFileName(i))
%         tic
%         disp(['Current solution attempt #' num2str(i)])
%         solDistribution = paramDistribution(numSamples,noiseAmt,...
```

```
%                                                    allData,pStar);
%         save(ansFileName(i),'solDistribution','noiseAmt','numSamples')
%
%         toc
%         diary off
%     end

disp('Finished FINALVcParams')

%=========================================================================
function solnDistribution
global computeAbsoluteError
fileNameAppend = 'VCdistribution_B7_';

diaryFileName = @(n)[fileNameAppend 'DiaryForEstimate' num2str(n) '.txt'];
ansFileName = @(n)[fileNameAppend 'VCDistributionSoln' num2str(n) '.mat'];
maxSamples = 1; %5
numEstimates = 100; %25

VcPlan = plans('VcholeraeOpt','numGam',1);
pWT = makeStrain([],{'WT V.cholerae'});
theSoln.X0 = stovec(pWT,VcPlan);
bestEst = theSoln.X0;

noiseAmt = 0.1;

addNoise = @(theData)( theData.*(1+noiseAmt*(2*rand(size(theData))-1) ) );

hold on
for i = 1:numEstimates
    diary(diaryFileName(i))
    tic
    disp(['Current solution attempt #' num2str(i)])
```

```
    [allData,p]= VCinitialize;

    allData.step2Data = addNoise(allData.step2Data);
    allData.hapRLevels = addNoise(allData.hapRLevels);
    allData.Fig7 = addNoise(allData.Fig7);

    [p, ERR, X, opt] = main(bestEst,allData);

    theSoln.X = X;
    theSoln.ERR = ERR;
    theSoln.ansNorm = norm(ERR);
    theSoln.opt = opt;
    theSoln.computeAbsoluteError = computeAbsoluteError;
    theSoln.allData = allData;

    cd('C:\..\MATLAB\VC Data\Vc Distributions')
    save(ansFileName(i),'theSoln')
    cd('C:\..\MATLAB\VC')

    theTime = toc/60;
    disp(['Running time: ' num2str(theTime) ' min'])
    diary off
end

%=========================================================================
function [bestNorm, bestEst, otherEst] = findBestEst(maxSamples,allData)
global fileNameAppend

paramPlan = plans('VcholeraeOpt','numGam',1);
numParams = getNumVarsInPlan(paramPlan);

sampleResult = mainToOpt(5*rand(numParams,1)+1,allData);
numData = length(makeVect(sampleResult));
```

```
otherEst.ERR = zeros(numData,maxSamples);
otherEst.ERRnorm = zeros(1,maxSamples);
otherEst.X = zeros(numParams,maxSamples);


for i = 1:maxSamples
    disp(['Working on estimate #' num2str(i)])
    otherEst.X(:,i) = generateRandEstimate('VCguessRanges.txt');
    otherEst.ERR(:,i) = mainToOpt(otherEst.X(:,i),allData);
    otherEst.ERRnorm(i) = norm(otherEst.ERR(:,i));
    save([fileNameAppend 'otherEstData.mat'],'otherEst')
end
[theMin, at] = min(otherEst.ERRnorm);
bestEst = otherEst.X(:,at);
bestNorm = theMin;

%=========================================================================
function varargout = main(X0,allData)
% Optimization parameters
maxIter = 10; % # of iterations/lsqnonlin iterate
maxEvals = 1000000;
tol = 1e-14;
genTol = 1e-15;
optimality = tol;
dispStr = 'iter';
ctrMax = 1; % # of repetitions of the lsqnonlin

LB = zeros(size(X0));
LB(25:28) = 1;
UB = inf*ones(size(X0));
UB(end) = 1;

[X,ERR,opt] = generalOptRoutine(@(y)mainToOpt(y,allData),X0,'maxIter',...
        maxIter,'maxEvals',maxEvals,'normTol',tol,'genTol',genTol,...
```

```
        'dispStr',dispStr,'optimality',optimality,'ctrMax',ctrMax,...
        'computeJacobian',true,'upper bound',UB,'lower bound',LB);

disp('Final answer found')
paramPlan = plans('VcholeraeOpt','numGam',1);
p = vectos(X,paramPlan);

if nargout > 0
    varargout{1} = p;
end
if nargout > 1
    varargout{2} = ERR;
end
if nargout > 2
    varargout{3} = X;
end
if nargout >3
    varargout{4} = opt;
end

%=========================================================================
function [f,df] = mainToOpt(X,allData)
paramPlan = plans('VcholeraeOpt','numGam',1);
p = vectos(X,paramPlan);

if nargout ==1
    ERR2 = makeTable1(p,'data',allData.step2Data,'allData',allData);
    ERR3A = makeFig6(p,'data',allData.hapRLevels,'allData',allData);
    ERR3B = makeFig7(p,'data',allData.Fig7,'allData',allData);
else
%     disp('Table 1')
 [ERR2, Diff2] = makeTable1(p,'data',allData.step2Data,'allData',allData);
%     disp('Fig 6')
[ERR3A, Diff3A] = makeFig6(p,'data',allData.hapRLevels,'allData',allData);
```

```
%     disp('Fig 7')
    [ERR3B, Diff3B] = makeFig7(p,'data',allData.Fig7,'allData',allData);

    df = [Diff2; ...
        Diff3A; ...
        Diff3B];
end
f = [makeVect(ERR2); ...
    makeVect(ERR3A); ...
    makeVect(ERR3B)];
```

B.2 Initialize the *V. cholerae* Optimization

```
function  [allData, p]= VCinitialize

allData.TableData = [5568768606 13975133803 661927655 7368254360
                     14880099371 37274606372 15864812309 37784865916
                     2827296191 15399782928 602901871 4694930889
                     3400395244 22618133039 2886491855 17857246552
                     3840124475 9265658740 384474828 9182292778
                     6624330372 18236619628 6164785959 27906352549];
load('Svenningsen2009Fig5_RawData.mat','Data')
allData.luxOAUCC = Data.Y(:,3)';

allData.step2Data = VCformStep2Data(allData.TableData,allData.luxOAUCC);

% Figure 6 from Svenningsen 2009
allData.hapRLevels = [6.3 2.3 2.3 1.1 35];

% Figure 7 from Svenningsen 2009
load('Svenningsen2009Fig7_RawData.mat','Data')
Fig7 = Data.Y;
Fig7(logical(Fig7<0.04)) = 0;
allData.Fig7 = Fig7(:,1:4);
```

```
allData.qrr4HCDFold = 2.9311e+007/3.5904e+006;

if nargout >= 2
    %%% Initialize Parameters %%%
    p = getParams('updateParams','numsRNA',4);
end
allData.residualKey = residualKey('species','Vc');
```

B.3 Model Experiment in Figure 3.7

```
function [f,df] = makeTable1(P,varargin)

toPlot = false;
onlyData = false;
weightAnswer = false;
for i = 1:2:length(varargin)
    switch lower(varargin{i})
        case lower('SS')
            WT = varargin{i+1};
        case lower('Data')
            data = varargin{i+1};
            withData = true; % Obsolete??
        case lower('Plot')
            toPlot = varargin{i+1};
        case lower('Data only')
            data = varargin{i+1};
            onlyData = true;
        case lower('allData')
            allData = varargin{i+1};
            weightAnswer = true;
        otherwise
            error('Unknown case')
    end
end
returnJac = false;
```

```
if nargout == 2
    returnJac = true;
end

if onlyData
    f = data;
    toPlot = true;
else
    if ~exist('WT','var')
        WT = getSSAt(P);
    end

    if ~returnJac
        promInfoWT = computePromInfo(P,P);
        promInfoHapR = computePromInfo(makeStrain(P,{'hapR mRNA'}),P);
    promInfoAUCC = computePromInfo(makeStrain(P,{'luxO-Qrr Feedback'}),P);
    else
        [promInfoWT dWT] = computePromInfo(P,P);
     [promInfoHapR dHapR]= computePromInfo(makeStrain(P,{'hapR mRNA'}),P);
        [promInfoAUCC dAUCC]= computePromInfo(makeStrain(P,{...
                                            'luxO-Qrr Feedback'}),P);
    end

    %6x4 same interpretation as the input for "TableData"
    prom = [promInfoWT promInfoHapR promInfoAUCC]';
    luxOAUCC = promInfoAUCC(:,2)./promInfoAUCC(:,1);
    f = VCformStep2Data(prom,luxOAUCC');

    if returnJac
        df = computeTable1Jac(prom,dWT,dHapR,dAUCC);
    end

    if exist('data','var')
        if returnJac
```

```
            [f df] = computeError(f,data,df);
        else
            f = computeError(f,data);
        end
    end
end

if toPlot
    bar(f(:,end-1),'group');
    set(gca,'XTickLabel',{'qrr1','qrr2','qrr3','qrr4'})
    xlabel('Reporter Fusion')
    ylabel('Fold Change in Luminesence')
    xlim([0 5])
    ylim([0 18])
end

if weightAnswer
    f = f*allData.residualKey.Table1.weight;
    if returnJac
        df = df*allData.residualKey.Table1.weight;
    end
end

%=========================================================================
function dF = computeTable1Jac(prom,dWT,dHapR,dAUCC)
[mp np] = size(prom);  %mp=6,np=4

dF = [dWT;dHapR;dAUCC];
[m,n] = size(dF);
J = dF; % <-- Reflects the jacobian of prom'
dF = zeros(m,n);

%%% VERSION A %%%
PROM = prom;
```

```
relToSS = PROM(4,1);
relToSen = dHapR(4+1,:);
prom = prom';
for j = 1:4 % qrr index
    for i = 1:mp-2 % experiment index
        startIdx = j+4*(i-1);
        dF(startIdx,:) = diffModel(relToSS,relToSen,prom(startIdx),...
                                   J(startIdx,:));
    end
    idx = mp-1;
    startIdx = j+4*(idx-1);
    dF(startIdx,:) = diffModel(PROM(end-1,j),dAUCC(j,:),PROM(end,j),...
                                   dAUCC(4+j,:));
end

%=========================================================================
function df = diffModel(ssRelTo,senRelTo,ss,sen)
df = 1/ssRelTo^2*(ssRelTo*sen-ss*senRelTo);

%=========================================================================
function labelPlot(p)
% Make labels for graph
xLabels = {};
legNames = {'WT','\Deltaqrr1-4','\DeltahapR','\DeltahapR,\Deltaqrr1-4',...
                        'luxO AUCC','luxO AUCC, \Deltaqrr1-4'};
for i = 1:p.numsRNA
    xLabels = { xLabels{:}, ['qrr' num2str(i)]};
end
set(gca,'XTickLabel',xLabels)
xlabel('Reporter Fusion')
ylabel('Luminesence')
legend(legNames,'Location','North')

%=========================================================================
```

```
function [promInfo Dprom]= computePromInfo(p,pWT) %,varargin)
% ****************************************
% Make sure that ss.r =0 in the hapR strain
% ****************************************

pMut = makeStrain(p,{'qrri RNA'},{[1:4]});
ssWT = getSSAt(p);
ssMut = getSSAt(pMut);

% if no hapR
if sum(logical(p.Er~=0))==0 && sum(logical(p.Kl~=0))==0 && ...
                            sum(logical(p.Vq~=0))==0
    ssWT.r = 0;
    ssMut.r = 0;
end

if nargout ==1
    promWT = VCpromoters(pWT,ssWT);
    promMut = VCpromoters(pWT,ssMut);
elseif nargout >=2 % && nargin == 2
    [promWT, Dwt] = VCpromoters(pWT,ssWT);
    [promMut, Dmut] = VCpromoters(pWT,ssMut);

    thePlan = plans('VcholeraeOptWStates','numGam',1);
    numParams = thePlan.numParams;

    temp = sensitivityStatesToParams(ssWT,p,thePlan);
    sigmaP = temp{1};

    temp = sensitivityStatesToParams(ssMut,pMut,thePlan);
    sigmaPmut = temp{1};

    numProms = size(Dwt,1);
    Dprom = zeros(2*numProms,numParams);
```

```
    %Dwt*SIG;
    Dprom(1:numProms,:) = Dwt(:,1:numParams) + ...
                          Dwt(:,numParams+1:end)*sigmaP;

    %Dmut*SIG;
    Dprom(1+numProms:2*numProms,:) = Dmut(:,1:numParams) + ...
                          Dmut(:,numParams+1:end)*sigmaPmut;
else
    error('Wrong input types in computePromInfo(p,varargin)')
end
promInfo = [promWT.q promMut.q];
```

B.4 Model Experiment in Figure 3.5

```
function [hapR,dhapR] = makeFig6(P,varargin)
global computeAbsoluteError
toPlot = false;
numArgs = 5;
onlyData = false;
weightAnswer = false;
knockOutStrains = @(n)[[1:n-1], [n+1:4]];
for i = 1:2:length(varargin)
    switch lower(varargin{i})
        case lower('SS')
            WT = varargin{i+1};
        case lower('Data')
            data = varargin{i+1};
            numArgs = length(data);
        case lower('Plot')
            toPlot = varargin{i+1};
        case lower('Data only')
            data = varargin{i+1};
            onlyData = true;
```

```
        case lower('allData')
            allData = varargin{i+1};
            weightAnswer = true;
        otherwise
            error('Unknown case')
    end
end
returnJac = false;
if nargout == 2
    returnJac = true;
end

if onlyData
    toPlot = true;
    hapR = data;
else
    if ~exist('WT','var')
        WT = getSSAt(P);
    end

    hapR = zeros(1,numArgs);
    mutSS = cell(1,numArgs+1);
    mutP = cell(1,numArgs+1);
    for i = 1:numArgs-1 % Insert one sRNA only
        mutP{i} = makeStrain(P,{'qrri RNA'},{knockOutStrains(i)});
        mutSS{i} = getSSAt(mutP{i});
        hapR(i) = mutSS{i}.r;
    end
    mutP{numArgs} = makeStrain(P,{'qrri RNA'},{[1:4]});
    mutSS{numArgs} = getSSAt(mutP{numArgs});
    hapR(end) = mutSS{numArgs}.r;

    mutP{numArgs+1} = P;
    mutSS{numArgs+1} = WT;
```

```
    hapR = hapR/mutSS{numArgs+1}.r;

    if returnJac
        dhapR = computeFig6Jac(mutSS,mutP);
    end

    if exist('data','var')
        if returnJac
            [hapR, dhapR] = computeError(hapR,data,dhapR);
        else
            hapR = computeError(hapR,data);
        end
    end

    if weightAnswer
        hapR = hapR*allData.residualKey.Fig6.weight;
        if returnJac
            dhapR = dhapR*allData.residualKey.Fig6.weight;
        end
    end
end

if toPlot
    bar(hapR)
    labelPlot(P)
end

%=========================================================================
function dF = computeFig6Jac(ssMut,mutP)
thePlan = plans('VcholeraeOptWStates','numGam',1);
numStrains = length(mutP);

% COMPUTE SENSITIVITIES
```

```
sigma = cell(numStrains,1);
for i = 1:numStrains
    temp = sensitivityStatesToParams(ssMut{i},mutP{i},thePlan);
    sigma{i} = temp{1};
end
dF = zeros(numStrains-1,length(sigma{1}(1,:)));
for i = 1:numStrains-1
    dF(i,:) = diffModel(ssMut{end},sigma{end},ssMut{i},sigma{i});
end

%=========================================================================
function df = diffModel(ssWT,senWT,ssMut,senMut)
rWT = ssWT.r;
df = 1/rWT^2*(senMut(1,:)*rWT-ssMut.r*senWT(1,:));

%=========================================================================
function labelPlot(p)
% Make labels for graph
xLabels = {};
for i = 1:p.numsRNA
    xLabels = { xLabels{:}, ['qrr' num2str(i)]};
end
xLabels = {xLabels{:}, '-qrr1-4'};
set(gca,'XTickLabel',xLabels)
xlabel('Qrr genotype')
if strcmpi(p.name,'V. cholerae')
    ylabel('Relative {\it hapR} mRNA Concentration')
else
    ylabel('Relative {\it luxR} mRNA Concentration')
end
```

B.5 Model Experiment in Figure 3.6

```
function [RNA,dRNA] = makeFig7(P,varargin)
global computeAbsoluteError
```

```
toPlot = false;
onlyData = false;
weightAnswer = false;
numsRNA = 4;
for i = 1:2:length(varargin)
    switch lower(varargin{i})
        case lower('SS')
            WT = varargin{i+1};
        case lower('Data')
            data = varargin{i+1};
        case lower('Plot')
            toPlot = varargin{i+1};
        case lower('Data only')
            data = varargin{i+1};
            onlyData = true;
        case lower('allData')
            allData = varargin{i+1};
            weightAnswer = true;
        otherwise
            error('Unknown case')
    end
end
if onlyData
    RNA = data;
    toPlot = true;
else
    if ~exist('WT','var')
        WT = getSSAt(P);
    end
    % Figure 7 progressive sRNA knockout strains
    params4Strains = cell(4,1);
    ss4Strains = cell(4,1);
    params4Strains{1} = P;
    ss4Strains{1} = WT;
```

```
    formRNA = @(ss)[ss.q; ss.r];
    RNA = zeros(5,numsRNA);
    relTo = formRNA(WT);
    RNA(:,1) = relTo./relTo;
    knockOutStrains = [3 2 1];
    for i = 1:3
        params4Strains{1+i} = makeStrain(P,{'qrri RNA'},...
                                        {knockOutStrains(1:i)});
        ss4Strains{1+i} = getSSAt(params4Strains{1+i});
        RNA(:,i+1) = formRNA(ss4Strains{1+i})./relTo;
    end
    RNA(logical(RNA<1e-13)) = 0;

    if exist('data','var')
        RNA = computeError(RNA,data);
        RNA(isnan(RNA)) = 0;
    end
end
if toPlot
    bar(RNA,'group');
    labelPlot
end

if weightAnswer
    RNA = RNA*allData.residualKey.Fig7.weight;
end

if nargout == 2
    dRNA = computeFig7Jac(params4Strains,ss4Strains);
    if exist('data','var') && ~computeAbsoluteError
        dRNA = scaleJacobianByData(dRNA,data);
    end
    if weightAnswer
```

```
        dRNA = dRNA*allData.residualKey.Fig7.weight;
    end
end

%=========================================================================
function dF = computeFig7Jac(params4Strains,ss4Strains)
everythingPlan = plans('VcholeraeOptWStates','numGam',1);
paramPlan = plans('VcholeraeOpt','numGam',1);
N = getNumVarsInPlan(paramPlan);
numRNA = 5;
numStrains = length(params4Strains);
dF = zeros(numStrains*numRNA,N);
sigma = cell(numStrains,1);

% Compute sensitivities
for i = 1:numStrains
    temp = sensitivityStatesToParams(ss4Strains{i},params4Strains{i},...
                                        everythingPlan);
    sigma{i} = temp{1};
end

% Compute derivative of the model
for i = 1:numStrains
    dF(1+numRNA*(i-1):i*numRNA,:) = diffModel(ss4Strains{1},sigma{1},...
                            ss4Strains{i},sigma{i},params4Strains{i});
end

%=========================================================================
function df = diffModel(ssWT,senWT,ssMut,senMut,p)
numStates = 5;
numParams = size(senWT,2);
df = zeros(numStates,numParams);

for i = 1:4
```

```
    if p.Kp(i) ~=0
        df(i,:) = (ssWT.q(i)*senMut(2+i,:)-...
                        ssMut.q(i)*senWT(2+i,:))./(ssWT.q(i))^2;
    end
end
df(end,:) = (ssWT.r*senMut(1,:)-ssMut.r*senWT(1,:))*1/(ssWT.r)^2;

%=========================================================================
function labelPlot
% Make labels for graph
numsRNA = 4;
xLabels = {};
legNames = {'WT','\Deltaqrr3','\Deltaqrr2,3','\Deltaqrr1,2,3'};
for i = 1:numsRNA
    xLabels = { xLabels{:}, ['qrr' num2str(i)]};
end
xLabels = {xLabels{:}, 'hapR'};
set(gca,'XTickLabel',xLabels)
xlabel('RNA')
ylabel('Relative RNA Concentration')
legend(legNames,'Location','NorthWest')
```

B.6 Format *V. cholerae* Data

```
function data = VCformStep2Data(prom,luxOAUCC)
[m,n] = size(prom);

%%% VERSION A %%%
%%% Normalize the data according to the original version.
relTo = prom(4,1);
prom(1:end-2,:) = prom(1:end-2,:)/relTo;
prom(end-1,:) = prom(end,:)./prom(end-1,:);
prom(end,:) = 1;
data = prom';
```

APPENDIX C

MATLAB CODE: SRNA MODEL

The following is a collection of the essential Matlab code used for the steady-state equations for the sRNA circuit.

C.1 Steady-state Map

```
function [G,DG] = steady_state_map(params,varargin)
% This is the steady-state ODE model for the sRNA circuit and is used to
% compute the steady-state solutions.
%
% params: A structure containing all of the parameters and states for a
% SINGLE Gamma/AI
% thePlan: Output from "plans.m" specifying the plan for the Jacobian.

if nargin == 2
    thePlan = varargin{1};
end

sp = params;
if isfield(params,'AI') && ~isempty(params.AI)
% If Gamma is a function of AI, then evaluate Gamma at that AI
    GAMMA = GamAI(params.AI,sp);
    sp.Gamma = GAMMA;
else
    sp.AI = [];
end

if length(sp.Gamma) >1
```

```
    error('Gamma can only be a length of at most 1.')
end

numsRNA = length(sp.Kp);
G = zeros(2+2*numsRNA,1);
j=1;

% dr/dt = 0
G(j) = sp.r0 + (1-sp.r0)/(1+(sp.K_R * sp.r)^2) - ( sp.H'*sp.Er + 1)*sp.r;
j = j+1;

% do/dt = 0
G(j) = 1/(1+sp.K_O*(1+sp.Gamma)*sp.o) - (sp.H'*sp.Eo +1)*sp.o;
j = j+1;

% dq/dt = 0
for i=1:numsRNA,
 G(j) = (sp.Kp(i) * sp.Gamma * sp.o) / (1 + sp.Kp(i) * sp.Gamma * sp.o)...
      * (1 + sp.Vq(i) * (sp.Kl(i) * sp.r)^2) / (1 + (sp.Kl(i) * sp.r)^2)...
      - (sp.Eq(i) * (1- sum(sp.H)) +1)*sp.q(i);
    j = j+1;
end;

% dH/dt = 0
for  i=1:numsRNA,
 G(j) = (1- sum(sp.H)) * sp.q(i) - (sp.Er(i) * sp.r + ...
            sp.Vor * sp.Eo(i) * sp.o ) * sp.Vr(i) * sp.H(i);
 j = j+1;
end;

if nargout == 2 && nargin == 2
    % Compute the Jacobian if asked
    DG = makejac(D_steady_state_map(params),thePlan);
end
```

C.2 Jacobian of the Steady-state Map

```
function DG = D_steady_state_map(sp)
% To compute the Jacobian of the steady-state map (DF(p) in the manuscript)
% sp: A structure containing parameters and (steady)states for multiple
% cell densities
%
% DG: 10x1 cell where the DG{i,1} entry is a structure whose fields are the
% variables/parameters for the i'th steady-state equation.
% i=1 -> dr/dt =0
% i=2 -> do/dt =0
% i=3:6 -> dq/dt =0
% i=7:10 -> dH/dt =0

numDensity = length(sp.r);

haveLuxR = 1;
haveQrr = logical(sp.Kp>0);
numsRNA = length(sp.Kp);

if isfield(sp,'AI') && ~isempty(sp.AI)
% If Gamma is a function of AI, the override the default definition
    [sp.Gamma diffGamma] = GamAI(sp.AI,sp);
    diffByAI = true;
else
    sp.AI = [];
    diffByAI = false;
end

DG = cell(2+2*numsRNA,1);
j=1;
% Gradient of dr/dt = 0
% G(j) = sp.r0 + (1-sp.r0)/(1+(sp.K_R * sp.r)^2) - (sp.H'*sp.Er + 1)*sp.r;
g = [];
```

```
g.r0 = 1 - 1./(1+(sp.K_R * sp.r)^2);

g.K_R = -(1-sp.r0)*2*sp.r.^2*sp.K_R./(1+(sp.K_R * sp.r).^2).^2...
                    *logical(sp.K_R>0);

% For Vh...
knownParams = getKnownParams;
g.R0 = g.K_R*knownParams.KRd;

g.r   = -(1-sp.r0)*2*sp.r*sp.K_R^2./(1+(sp.K_R * sp.r).^2).^2 - ...
                ( (sp.H'*sp.Er)' + 1);
g.H   = -sp.Er*sp.r;
g.H(~haveQrr,:)=0; % Derivative =0 if q=0



g.Er  = -sp.H*diag(sp.r);
g.Er(logical(sp.Er==0),:) = 0; % Derivative =0 if Er=0
g.Er(~haveQrr,:) = 0; % Derivative =0 if q=0

DG{j} = g;
j = j+1;

% Gradient of do/dt = 0
%G(j) = 1/(1+sp.K_O*(1+sp.Gamma)*sp.o) - (sp.H'*sp.Eo +1)*sp.o;
g = [];

g.K_O = -(1+sp.Gamma).*sp.o./(1+sp.K_O*(1+sp.Gamma).*sp.o).^2...
                            *logical(sp.K_O>0);
dGamma = -sp.K_O*sp.o./(1+sp.K_O*(1+sp.Gamma).*sp.o).^2;
if diffByAI
    g.alpha = dGamma.*diffGamma;
else % if Gamma is not a function of AI...
    g.Gamma =  dGamma;
end
```

```
g.o = -sp.K_O*(1+sp.Gamma)./(1+sp.K_O*(1+sp.Gamma).*sp.o).^2 - ...
                        ( (sp.H'*sp.Eo)' +1);
g.H = -sp.Eo*sp.o;
g.H(~haveQrr,:)=0;

g.Eo = -sp.H*diag(sp.o);
g.Eo(logical(sp.Eo==0),:) = 0; % Derivative =0 if Eo=0
g.Eo(~haveQrr,:)=0;

DG{j} = g;
j = j+1;

% Gradient of dq/dt = 0
for i=1:numsRNA,
    g = [];
    if haveQrr(i)
%G(j) = (sp.Kp(i) * sp.Gamma * sp.o) / (1 + sp.Kp(i) * sp.Gamma * sp.o)...
%  *(1 + sp.Vq(i) * (sp.Kl(i) * sp.r)^2) / (1 + (sp.Kl(i) * sp.r)^2)...
%  - (sp.Eq(i) * (1- sum(sp.H)) +1)*sp.q(i);
    g.Kp =  zeros(numsRNA,numDensity);
    g.Kp(i,:) = (1 + sp.Vq(i) * (sp.Kl(i) * sp.r).^2) ./ (1 + (sp.Kl(i)...
                * sp.r).^2) .* sp.Gamma.* sp.o./(1 + sp.Kp(i)...
                    * sp.Gamma .* sp.o).^2;

      dGamma = (1 + sp.Vq(i) * (sp.Kl(i) * sp.r).^2) ./ (1 + ...
            (sp.Kl(i) * sp.r).^2).* sp.Kp(i) .* sp.o ./ (1 + ...
            sp.Kp(i) * sp.Gamma .* sp.o).^2;
        if diffByAI
            g.alpha = dGamma.*diffGamma;
        else
            g.Gamma =  dGamma;
        end
```

```
    g.o = (1 + sp.Vq(i) * (sp.Kl(i) * sp.r).^2) ./ (1 + (sp.Kl(i)...
          * sp.r).^2).* sp.Kp(i) .* sp.Gamma ./ (1 + sp.Kp(i) * ...
          sp.Gamma .* sp.o).^2;

    g.Vq    = zeros(length(sp.Vq),numDensity);
    g.Vq(i,:) = (sp.Kp(i) * sp.Gamma .* sp.o) ./ (1 + sp.Kp(i) * ...
          sp.Gamma .* sp.o) .* (sp.Kl(i) * sp.r).^2 ./ (1 + (sp.Kl(i)...
          * sp.r).^2)*logical(sp.Vq(i)~=1 || sp.Vq(i)~=0);

    g.Kl    = zeros(numsRNA,numDensity);
    g.Kl(i,:) = (sp.Kp(i) * sp.Gamma .* sp.o) ./ (1 + sp.Kp(i) * ...
          sp.Gamma .* sp.o)*2*sp.Kl(i).*sp.r.^2 * (sp.Vq(i)-1)./...
          (1 + (sp.Kl(i) * sp.r).^2).^2 ;

  % For Vh...
    g.R0 = g.Kl(i)*knownParams.Kl_prop(i)*haveQrr(i);


    g.r = (sp.Kp(i) * sp.Gamma .* sp.o) ./ (1 + sp.Kp(i) * ...
          sp.Gamma .* sp.o)*2*sp.Kl(i)^2.*sp.r * (sp.Vq(i)-1)...
          ./(1 + (sp.Kl(i) * sp.r).^2).^2*haveLuxR;

    g.Eq    = zeros(numsRNA,numDensity);
    g.Eq(i,:) = -(1- sum(sp.H)).*sp.q(i,:);

    g.H      = sp.Eq(i)*ones(4,numDensity)*diag(sp.q(i,:));
    g.H(~haveQrr,:) = 0;

    g.q      = zeros(numsRNA,numDensity);
    g.q(i,:)   = -(sp.Eq(i) * (1- sum(sp.H)) +1);
    g.q(~haveQrr,:) = 0;

  end
DG{j} = g;
```

```
  j = j+1;
end;

% Gradient of dH/dt = 0
for  i=1:numsRNA,
 %G(j) = (1- sum(sp.H)) * sp.q(i) - (sp.Er(i) * sp.r + sp.Vor * ...
 %                  sp.Eo(i) * sp.o ) * sp.Vr(i) * sp.H(i);
 g = [];
 if haveQrr(i)
     g.H  = -ones(numsRNA,numDensity)*diag(sp.q(i,:));
     g.H(i,:) = g.H(i,:) - (sp.Er(i)*sp.r+sp.Vor*sp.Eo(i)*sp.o)* sp.Vr(i);
     g.H(~haveQrr,:) = 0;

     g.q    = zeros(numsRNA,numDensity);
     g.q(i,:) = (1- sum(sp.H));
     g.q(~haveQrr,:) = 0;

     g.Er = zeros(numsRNA,numDensity);
     g.Er(i,:) =  - sp.r * sp.Vr(i) .* sp.H(i,:).*logical(sp.Er(i)>0);

     g.r   =  - sp.Er(i) * sp.Vr(i) * sp.H(i,:)*haveLuxR;

     g.Vor =  - sp.Eo(i) * sp.o * sp.Vr(i) .* sp.H(i,:)*logical(sp.Vor>0);

     g.Eo    = zeros(numsRNA,numDensity);
     g.Eo(i,:) = -sp.Vor*sp.o * sp.Vr(i) .* sp.H(i,:)*logical(sp.Eo(i)>0);

     g.o =  - sp.Vor * sp.Eo(i) * sp.Vr(i) * sp.H(i,:);

     g.Vr = zeros(numsRNA,numDensity);
     g.Vr(i,:) = - (sp.Er(i) * sp.r + sp.Vor * sp.Eo(i) * sp.o ) .* ...
                           sp.H(i,:)*logical(sp.Vr(i)>0);
 end
 DG{j} = g;
```

```
 j = j+1;
end;
```

C.3 Form the Jacobian

```
function J = makejac(g,thePlan)
% g: Output from D_steady_state_map(sp)
% (a cell whose entries are vectors representing the derivative of
% the state variables with respect to all of the parameters/states)
% thePlan: A structure that matches the output from "plans.m". Note that
% this structure can be different than the structure used to calculate the
% other input "g".
%
% J: A matrix representing the Jacobian of the system with respect to
% the variables/states outlined in "thePlan.plan". J(i,j) is the derivative
% of the derivative of the i'th parameter/variable in "thePlan.plan" with
% respect to the j'th parameter/variable in "thePlan.plan"

% total number of variables in this plan
N = getNumVarsInPlan(thePlan);

J = zeros(length(g),N);

for i=1:length(g),
 J(i,:) = stovec(g{i},thePlan)';
end;
```

C.4 Compute the Sensitivity of the States to the Parameters

```
function sigma = sensitivityStatesToParams(s,p,thePlan)
% To compute ds/dp. Let F(s,p) be the steady-state map, s the states
% and p the parameters. DF_s is the submatrix of the Jacobian where
% the states are perturbed, while DF_p is the submatrix of the Jacobian
% where the parameters are perturbed.
%
% Note: F(s,p) ~ F(s_0,p_0) + DF_s(s_0,p_0)ds + DF_p(s_0,p_0)dp
```

```
% At s=s_0, p=p_0, we have
% 0 = DF_s(s_0,p_0)ds + DF_p(s_0,p_0)dp
% --> ds/dp = DF_p(s_0,p_0)/DF_s(s_0,p_0)

% s and p are structures representing the steady states (at each cell
% density) and the parameters
% planName is the plan to follow to compute the Jacobian
%
% Returns an nx1 cell sigma where M = sigma{i,1} is a matrix of the
% derivative of the states with respect to parameters at the i'th cell
% density. Furthermore, M(j,k) is the derivative of the j'th state with
% respect to the k'th parameter.

numDensity = length(s.r);

sigma = cell(numDensity,1);
P = p;
S = s;

GammaStartsAt = findVariable(thePlan,'Gamma1');
if isempty(GammaStartsAt)
    GammaStartsAt = findVariable(thePlan,'Gamma');
end

GammaEndsAt = GammaStartsAt+thePlan.numGam -1;

% Change the plan definition so that I don't get an out of bounds error
if ~thePlan.fnOfAI
    thePlan.plan{strmatch('Gamma',{thePlan.plan{:,1}}),2} = 1;
end

thePlan.plan{strmatch('q',{thePlan.plan{:,1}}),2} = 4;
thePlan.plan{strmatch('H',{thePlan.plan{:,1}}),2} = 4;
thePlan.plan{strmatch('r',{thePlan.plan{:,1}},'exact'),2} = 1;
```

```
thePlan.plan{strmatch('o',{thePlan.plan{:,1}}),2} = 1;

for i = 1:numDensity
    if thePlan.fnOfAI
        P.AI = p.AI(i);
    else
        gammaFiller = zeros(10,thePlan.numGam);
        P.Gamma = p.Gamma(i);
    end

    S.r = s.r(i);
    S.o = s.o(i);
    S.q = s.q(:,i);
    S.H = s.H(:,i);

    sp = mergeStructures(P,S);
    DF = makejac(D_steady_state_map(sp),thePlan);

    if ~thePlan.fnOfAI
        gammaFiller(:,i) = DF(:,GammaStartsAt);
        DF = [DF(:,1:GammaStartsAt-1) gammaFiller ...
                    DF(:,GammaStartsAt+1:end)];
    end
    numStates = 10;
    DFDP = DF(:,1:end-numStates);
    DFDS = DF(:,end-numStates+1:end);
    % Rows that have at least one non-zero
    inCell = logical(sum(DFDS,2)~=0);

    if sum(~inCell) == 0 % if there are no knockouts
        Z = -DFDS\DFDP;
    else
        Z = -pinv(DFDS)*DFDP;
    end
```

```
    sigma{i} = Z;
end
```

APPENDIX D

MATLAB CODE: OPTIMIZATION FUNCTIONS

The following is a collection of essential Matlab code common to the *V. harveyi* and *V. cholerae* parameterizations.

D.1 Solve for the Steady-State

```
function ss = getSSAt(p,varargin)
% Returns the steady-state solution for the given parameterization
% [r o q H]

% Want to find the steady-states, knowing the parameters
planStr = 's';
numsRNA = length(p.Kp);

if isfield(p,'AI') && ~isempty(p.AI)
    cellDensityRange = p.AI;
    thePlan = plans(planStr,'AI',cellDensityRange(1),'numsRNA',numsRNA);
    fnOfAI = true;
else
    cellDensityRange = p.Gamma;
    thePlan = plans(planStr,'numGam',1,'numsRNA',numsRNA);
    fnOfAI = false;
end
N = getNumVarsInPlan(thePlan);
Y0 = 0.1*rand(N,1);
if nargin >1
    for i = 1:2:length(varargin)-1
```

```
        switch lower(varargin{i})
            case lower('initGuess')
                Y0 = varargin{i+1};
            otherwise
                error('Wrong input option in getSSAt')
        end
    end
end

maxEvals = 1000000;
maxIter = 17000; %700; %1.75*700;
normTol = 1e-12;
genTol = 1e-12;
dispStr = 'off';
ctrMax = 8;
computeJacobian = true;

for i=1:length(cellDensityRange)

    if fnOfAI
        thePlan = plans(planStr,'AI',cellDensityRange(i));
        p.AI = cellDensityRange(i);
        p.Gamma = [];
    else
        p.Gamma = cellDensityRange(i);
        p.AI = [];
    end
    x0 = Y0;
    noQrrAt = logical(p.Kp==0);
    if sum(noQrrAt)>0
        stemp = vectos(x0,thePlan);
        stemp.q(noQrrAt,:) = 0;
        stemp.H(noQrrAt,:) = 0;
        x0 = stovec(stemp,thePlan);
```

```
    end
    Y = generalOptRoutine(@(x)steady_state_map(vectos(x,thePlan,p),...
        thePlan),x0,'maxEvals',maxEvals,'maxIter',maxIter,...
        'genTol',genTol,'dispStr',dispStr,'normTol',normTol,...
        'ctrMax',ctrMax,'computeJacobian',computeJacobian);
    Y(logical(Y<=normTol*5))=0;
    % Make a structure of steadystate values for r o q and H and assign
    % them to the output vector.
    SS = vectos(Y,thePlan,p);
    ss.r(i) = SS.r;
    ss.o(i) = SS.o;
    ss.q(:,i) = SS.q;
    ss.H(:,i) = SS.H;
end
```

D.2 Optimization Routine

```
function [X,varargout] = generalOptRoutine(fHandle,X0,varargin)
% An overly complicated function to run lsqnonlin with different
% combinations of options. Overtime, on further development of the
% parameterization code, there was no need to change the optimization
% options.

% options = optimset('MaxFunEvals',maxEvals,'MaxIter',maxIter,...
%   'TolX',genTol,'TolFun',genTol,'Display',dispStr,'DiffMinChange',...
%   diffTol,'DiffMaxChange',diffTolMax);

global fileNameAppend
global computeAbsoluteError

lb = zeros(size(X0));
ub = inf*ones(size(X0));

options = optimset;
```

```
ctrMax = 1;
normTol = 1e-5;
for i = 1:2:length(varargin)-1
    switch lower(varargin{i})
        case lower('maxIter')
            options = optimset(options,'maxIter',varargin{i+1});
        case lower('maxEvals')
            options = optimset(options,'MaxFunEvals',varargin{i+1});
        case lower('normTol')
            normTol = varargin{i+1};
            options = optimset(options,'TolFun',normTol);
        case lower('genTol')
            options = optimset(options,'TolX',varargin{i+1});
        case lower('dispStr')
            options = optimset(options,'Display',varargin{i+1});
        case lower('ctrMax')
            ctrMax = varargin{i+1};
        case lower('upper bound')
            ub = varargin{i+1};
        case lower('lower bound')
            lb = varargin{i+1};
        case lower('computeJacobian')
            if varargin{i+1}
                options = optimset(options,'Jacobian','on');
            else
                options = optimset(options,'Jacobian','off');
            end
        case lower('givenJacobianPattern')
            options = optimset(options,'JacobPattern',varargin{i+1});
        otherwise
            error(['Unknown option in generalOptRoutine: ' varargin{i}])
    end
end
```

```
optimset('MaxFunEvals',maxEvals,'MaxIter',maxIter,'TolX',genTol,...
        'TolFun',genTol,'Display',dispStr,'JacobPattern',JacPat);
filePath = @(n)[fileNameAppend 'generalOptRoutine_Result_' num2str(n) '.mat'];

oldRes = 1e10;
resnorm = inf;
ctr = 0;
opt = 1e10;
X = X0;
while resnorm> normTol && ctr < ctrMax && resnorm ~= oldRes
    computeAbsoluteError = false;
    oldRes = resnorm;
    [X,resnorm,residual,exitflag,output] = lsqnonlin(...
                                                fHandle,X,lb,ub,options);
    opt = output.firstorderopt;
    ctr = ctr + 1;
end
err = fHandle(X);
if nargout > 1
    varargout{1} = err;
end
if nargout > 2
    varargout{2} = opt;
end
if nargout > 3
    varargout{3} = ctr;
end
```

APPENDIX E

MATLAB CODE: UTILITY FUNCTIONS

The following is a collection of the essential utility functions common to the *V. harveyi* and *V. cholerae* parameterization.

E.1 Vector to Structure

```
function s = vectos(v,thePlan,varargin)
% To transform a vector, v, into a structure, s, given a plan for doing it
%
% 1) Assign the components of v according to thePlan.
% 2) Assign any other known parameters according to the contents of
% varargin{1} (i.e. if you want to find the steady states, v = ss guess,
% varargin{1} = known parmeters)
%
% Returns a structure whose field names are the steadystates (at all cell
% densities) and parameters (incl all Gamma's).

s = fillOutS(v,thePlan);
if nargin >2
    p = varargin{1};
end
switch lower(thePlan.name)
    case lower({'VhOpt','VhOptStep2Jack','Testing','TestingWithStates',...
                    'VhTest','VhTestParams'})
        % Need to override these definitions with a "makeStrain" call in
        % the event that there's a LuxR-Auto and/or RQ feedback mutant.

        knownParams = getKnownParams;
```

```
        s.K_R = knownParams.KRd*s.R0;
        s.Kl = knownParams.Kl_prop*s.R0;

        s.r0 = knownParams.r0;
        s.Vq = [0; s.Vq];
    case lower('params')
        % Guessing the parameters only, so fill in the states
        if exist('p','var')
            if isstruct(p)
                s = mergeStructures(p,s);
            else
                thePlan = plans('s');
                s = fillOutS(p,thePlan.plan,s);
            end
        end
    case lower('states')
        % Guessing the states only, so fill in the parameters
        if exist('p','var')
            if isstruct(p)
                s = mergeStructures(p,s);
            else
                thePlan = plans('p');
                s = fillOutS(p,thePlan.plan,s);
            end
        end

    case lower({'VcOpt','VcholeraeOpt','VcholeraeOptWStates'})
        if exist('p','var')
            s = mergeStructures(p,s);
        end
end
s.name = thePlan.name;
if thePlan.fnOfAI
    s.AI = thePlan.AI;
```

```
end

%==========================================================================
function s = fillOutS(v,thePlan,varargin)
% # of sRNA = 4 only.
name = thePlan.name;
plan = thePlan.plan;

if nargin == 2
    s = [];
else
    s = varargin{1};
end
j=1;
for i=1:length(plan),
    n = plan{i,2}; % length of the data for field name in plan{i,1}
    if sum(strcmpi(plan{i,1},{'H','q'})) && n == 4
        % Form a 4 x n/4 matrix
        s.(plan{i,1}) = reshape(v(j + (0:n-1)'),4,n/4);

    elseif sum(strcmpi(plan{i,1},{'r','o','Gamma','AI'}))
        % Form a 1 x n vector (rather than an n x 1 vector)
        s.(plan{i,1}) = v(j + (0:n-1)')';

    else
        try
            s.(plan{i,1}) = v(j + (0:n-1)');
        catch
            if strcmp(plan{i,1},'r0')
                s.(plan{i,1}) = 0;
            else
                error('Something is wrong.')
            end
        end
```

```
    end
    j = j + n;
end;
```

E.2 Structure to Vector

```
function v = stovec(sp,thePlan)
% Performs the inverse operation of vectos.
% Transforms a structure s with fields scalars or vectors
% into a big vector. The order is alphabetical with the field names

plan = thePlan.plan;
j = 1;
N = getNumVarsInPlan(thePlan);
v = zeros(N,1);
for i=1:length(plan),
    n = plan{i,2};
    if (isfield(sp,plan{i,1}))
        copyThis = makeVect(sp.(plan{i,1}));
        if length(copyThis) == n
              v(j + (0:n-1)) = copyThis;
        elseif length(copyThis) == 2
            error('This case is not handled in this function.')
        elseif length(copyThis) == n +1
              Vq = copyThis;
              v(j + (0:n-1)) = Vq(2:end);
        end
    else
        v(j + (0:n-1)) = zeros(n,1);
    end;
    j = j+n;
end;
```

E.3 Parameterization Plans

```
function thePlan = plans(str,varargin)
% A structure containing the "plan" (i.e. the combination of parameters
```

```
% and variables one is interested in) along with other attributes of
% the system

i = 1;
%by default, assume there is at least one gamma and no AI.
numGam = 1;
fnOfAI = false;
numAI = 0;
AI = [];
numsRNA = 4; % Default value
while i<=length(varargin)-1
    switch lower(varargin{i})
        case lower('numGam')
            numGam = varargin{i+1}; % override numGam
            i = i+1;
        case lower('AI')
            AI = varargin{i+1};
            numAI = length(AI);
            fnOfAI = true;
            i = i+1;
        % IF you want to override the default value for the number
        % of sRNA in the strain
        case lower('numsRNA')
            numsRNA = varargin{i+1};
            i = i+1;
        otherwise
            error('Unknown option in plans(str,varargin)')
    end
    i = i+1;
end

if numGam+numAI == 0
 error('Need to have at least one Gamma or one AI in the plan definition')
end
```

```
if numAI >0
    numDensity = numAI;
else
    numDensity = numGam;
end

testing = { 'R0', 1
            'Eo' , numsRNA
            'Eq',  numsRNA
            'Er' , numsRNA
            'Gamma', numGam % <-- different than Vh params
            'K_O', 1
            'Kp', numsRNA
            'Vor', 1
            'Vq', max([numsRNA-1,1])
            'Vr', numsRNA};

VhParams = { 'R0', 1
            'Eo' , numsRNA
            'Eq',  numsRNA
            'Er' , numsRNA
            'Gamma', numGam
            'K_O', 1
            'Kp', numsRNA
            'Vor', 1
            'Vq', max([numsRNA-1,1])
            'Vr', numsRNA};

VhParamsTest = { 'R0', 1
            'Eo' , numsRNA
            'Eq',  numsRNA
            'Er' , numsRNA
            'Gamma', numGam
```

```
            'K_O', 1
            'Kp', numsRNA
            'Vor', 1
            'Vq', max([numsRNA-1,1])
            'Vr', numsRNA};

states = {'r', 1*numDensity
          'o', 1*numDensity
          'q', numsRNA*numDensity
          'H', numsRNA*numDensity};

% Does not have R0. Use this with Vc optimization
generalParams =  {'Eo' , numsRNA
            'Eq',  numsRNA
            'Er' , numsRNA
            'Gamma',numGam
            'Kl', numsRNA
            'K_O', 1
            'Kp', numsRNA
            'K_R', 1
            'Vor', 1
            'Vq', numsRNA
            'Vr', numsRNA
            'r0',1};
dependentVars = 2+2*numsRNA;
switch lower(str)
    case 'p'
        aPlan = generalParams;
        planName = 'Params';
        numStates = 0;
    case 's'
        aPlan = states;
        planName = 'States';
        numStates = dependentVars*numDensity;
```

```
case 'ps'
    aPlan = mergePlans(generalParams,states);
    planName = 'ParamsAndStates';
    numStates = dependentVars*numDensity;

case lower('VcholeraeOpt')
    aPlan = generalParams;
    planName = 'VcholeraeOpt';
    numStates = 0;

case lower('VcholeraeOptWStates')
    aPlan = mergePlans(generalParams,states);
    planName = 'VcholeraeOptWStates';
    numStates = dependentVars*numDensity;

case lower('VhOpt')
    aPlan = VhParams;
    planName = 'VhOpt';
    numStates = 0;

case lower('testing')
    aPlan = testing;
    planName = 'Testing';
    numStates = 0;

case lower('TestingWithStates')
    aPlan = mergePlans(testing,states);
    planName = 'TestingWithStates';
    numStates = dependentVars*numDensity;

case lower('VhOptStep2Jack')
    aPlan = mergePlans(VhParams,states);
    planName = 'VhOptStep2Jack';
```

```
        numStates = dependentVars*numDensity;

    case lower('VhTestAll')
        aPlan = mergePlans(VhParamsTest,states);
        planName = 'VhTest';
        numStates = dependentVars*numDensity;

    case lower('VhTestParams')
        aPlan = VhParamsTest;
        planName = 'VhTestParams';
        numStates = dependentVars*numDensity;

    otherwise
        error('Wrong string input argument.')
end

thePlan.numDensity = numDensity;
thePlan.AI = AI;
thePlan.fnOfAI = fnOfAI;
thePlan.numGam = numGam;
thePlan.numAI = numAI;
thePlan.plan = aPlan;
thePlan.name = planName;
thePlan.numStates = numStates;
numParams = getNumVarsInPlan(thePlan) - numStates;
thePlan.numParams = numParams*logical(numParams>0);
thePlan.numVars = numParams+numStates;


%=========================================================================
function newPlan = mergePlans(planA,planB)
[m,n] = size(planB);

newPlan = planA;
```

```
for i =1:m
    newPlan(end+1,:) = planB(i,:);
end
```

E.4 Merge Structures

```
function S = mergeStructures(from,into)
% from and into are structures. The output is the merger of the two
% structures so that "into" = "from" + "into"

S = into;
theNames = fieldnames(from);
for i = 1:length(theNames)
    S.(theNames{i}) = from.(theNames{i});
end
```

E.5 Number of Variables in a Plan

```
function N = getNumVarsInPlan(thePlan)
N = sum(cell2mat(thePlan.plan(:,2)));
```

E.6 Find Variable Index

```
function idx = findVariable(thePlan,varStr)
% Returns the index of the variable indicated by varStr in thePlan
names = varnames(thePlan.plan);
idx = strmatch(varStr,names,'exact');
```

E.7 Variable Name and Index for a Plan

```
function names = varnames(plan,varargin)
% gives variable name and index using a plan
j=1;
for i=1:length(plan),
    n = plan{i,2}; % length of the data for field name in plan{i,1}
    if n > 1
        for k=1:n,
        names{j + k -1} = sprintf('%s%d',plan{i,1},k);
        end
```

```
    else
         names{j} = sprintf('%s',plan{i,1});
    end
    j = j + n;
end

if nargin==2
    names = names{varargin{1}};
end
```

E.8 Apply Different Weights to Experiments

```
function key = residualKey(varargin)
% To apply different weights to individual batches of experiments
%
% key for the experimental data
speciesName = 'Vh';
for i =1:2:length(varargin)
    switch varargin{i}
        case lower('species')
            speciesName = varargin{i+1};
        case lower('states')
            states = varargin{i+1};
        otherwise
            error('Unknown option in residualKey')
    end
end

% If you're doing a Vh strain...
if strcmpi(speciesName,'Vh')
    key.step1Error.weight = 0; %1e2;
    key.step2Error.weight = 10; %1e1;
    key.step3Error.weight = 4; %1; %1;
    key.step4Error.weight = 1; %1e-1;
    key.step5Error.weight = 0;
```

```
else % If you're doing a Vc strain...
    key.Fig6.weight = 1;
    key.Fig7.weight = 1;
    key.Table1.weight = 1;
end
```

E.9 Make a Mutant Strain

```
function p = makeStrain(params,strainType,varargin)
% "Engineer" a new strain as specified in strainType (a cell), which can
% take on the following forms:
% 'hapR mRNA'
% 'luxOAUCC', 'luxO-Qrr Feedback'
% 'qrri RNA'
% 'delta qrri'
% 'hapR-Qrri'
% 'hapR-Qrr Feedback', 'luxR-Qrr Feedback'
% 'luxO-Qrri'
% 'hapR Auto'
% 'luxO Auto'
% 'WT V.harveyi'
% 'WT V.cholerae'
% 'identical qrr'
% 'distribution'
% 'at LCD', 'at HCD'
% 'no hapR repression'
% Varargin specifies to which sRNA you are applying the knockout to and is
% a cell of same size as strainType
optCtr = 1;
p = params;
for i = 1:length(strainType)
    aStrain = lower(strainType{i});
    switch aStrain
        case lower({'hapR mRNA','luxR mRNA'})
            p.Er(:) = 0;
```

```
    p.Kl(:) = 0;
    p.Vq(:) = 0;
    p.K_R = 0;
    p.R0 = 0;

case lower({'luxOAUCC','luxO-Qrr Feedback'})
    p.Eo(:) = 0;
    p.Vor = 0;

case lower('luxO-Qrri')
    these = varargin{1}{optCtr};
    p.Eo(these) = 0;
    optCtr = optCtr +1;

case lower({'delta qrri','qrri RNA'})
% removes qrr but doesn't shrink the array
    these = varargin{1}{optCtr};
    p.Kp(these) = 0;
    p.Er(these) = 0;
    p.Eo(these) = 0;
    p.Eq(these) = 0;
    p.Kl(these) = 0;
    p.Vq(these) = 0;
    p.Vr(these) = 0;
    optCtr = optCtr +1;

case lower({'hapR-Qrri','luxR-Qrri'})
    these = varargin{1}{optCtr};
    p.Kl(these) = 0;
    p.Vq(these) = 0;
    optCtr = optCtr +1;

case lower({'hapR-Qrr Feedback', 'luxR-Qrr Feedback'})
    p.Kl(:) = 0;
```

```
    p.Vq(:) = 0;

case lower({'hapR Auto','luxR Auto'})
    p.K_R = 0;

case lower('luxO Auto')
    p.K_O = 0;

case lower({'WT V.harveyi'})
    load('C:\..\MATLAB\SOLUTIONS\ANSWER_Vh.mat')
    p.name = ['V. ' aStrain(6:end)];
case lower({'WT V.cholerae'})
    load('C:\..\MATLAB\SOLUTIONS\ANSWER_Vc.mat')

case lower('WT')
    if isempty(params)
        error('Need to specify a parameter structure.')
    end
    p = params;

case lower('identical qrr')
    qrrIdx = varargin{1}{optCtr};
    p.Eq(:) = p.Eq(qrrIdx);
    p.Eo(:) = p.Eo(qrrIdx);
    p.Er(:) = p.Er(qrrIdx);
    p.Kp(:) = p.Kp(qrrIdx);
    p.Kl(:) = p.Kl(qrrIdx);
    p.Vq(:) = p.Vq(qrrIdx);
    p.Vr(:) = p.Vr(qrrIdx);
    optCtr = optCtr +1;

case lower('mean')
    p.Eq(:) = mean(p.Eq);
    p.Eo(:) = mean(p.Eo);
```

```
            p.Er(:) = mean(p.Er);
            p.Kp(:) = mean(p.Kp);
            p.Kl(:) = mean(p.Kl);
            p.Vq(:) = mean(p.Vq);
            p.Vr(:) = mean(p.Vr);

        case lower({'mean with distribution','distribution'})
            p.Eq(:) = abs(mean(p.Eq) + std(p.Eq)*randn(length(p.Eq),1));
            p.Eo(:) = abs(mean(p.Eo) + std(p.Eo)*randn(length(p.Eo),1));
            p.Er(:) = abs(mean(p.Er) + std(p.Er)*randn(length(p.Er),1));
            p.Kp(:) = abs(mean(p.Kp) + std(p.Kp)*randn(length(p.Kp),1));
            p.Kl(:) = abs(mean(p.Kl) + std(p.Kl)*randn(length(p.Kl),1));
            p.Vq(:) = abs(mean(p.Vq) + std(p.Vq)*randn(length(p.Vq),1));
            p.Vr(:) = abs(mean(p.Vr) + std(p.Vr)*randn(length(p.Vr),1));
        case lower({'Eo scale'})
            scaleFactor = varargin{1}{optCtr};
            p.Eo(:) = p.Eo(:)*scaleFactor;
            p.Vor = p.Vor/scaleFactor;
            optCtr = optCtr +1;

        case lower({'Er scale'})
            scaleFactor = varargin{1}{optCtr};
            p.Er(:) = p.Er(:)*scaleFactor;
            p.K_R = p.K_R*scaleFactor;
            p.Vr(:) = p.Vr(:)/scaleFactor;
            p.Vor = p.Vor*scaleFactor;
            p.Kl(:) = p.Kl(:)*scaleFactor;
            optCtr = optCtr +1;

        otherwise
            error('Unknown strain type')
    end
end
```

REFERENCES

[1] K. Anguige, J. King, and J. Ward, *Modelling antibiotic- and anti-quorum sensing treatment of a spatially-structured pseudomonas aeruginosa population*, Journal of Mathematical Biology, 51 (2005), pp. 557–594.

[2] ——, *A multi-phase mathematical model of quorum sensing in a maturing pseudomonas aeruginosa biofilm*, Mathematical Biosciences, 203 (2006), pp. 240–276.

[3] K. Anguige, J. King, J. Ward, and P. Williams, *Mathematical modelling of therapies targeted at bacterial quorum sensing*, Mathematical Biosciences, 192 (2004), pp. 39–83.

[4] J. F. Apgar, D. K. Witmer, F. M. White, and B. Tidor, *Sloppy models, parameter uncertainty, and the role of experimental design*, Molecular bioSystems, 6 (2010), pp. 1890–1900. PMID: 20556289 PMCID: PMC3505121.

[5] C. Baker, T. Jia, and R. V. Kulkarni, *Stochastic modeling of regulation of gene expression by multiple small RNAs*, Physical Review E, 85 (2012), p. 061915.

[6] S. Bandara, J. P. Schloder, R. Eils, H. G. Bock, and T. Meyer, *Optimal experimental design for parameter estimation of a cell signaling model*, PLoS Computational Biology, 5 (2009). PMID: 19911077 PMCID: PMC2775273.

[7] S. K. Banik, A. T. Fenley, and R. V. Kulkarni, *A model for signal transduction during quorum sensing in vibrio harveyi*, Physical Biology, 6 (2009), p. 046008.

[8] J. P. Bardill, X. Zhao, and B. K. Hammer, *The vibrio cholerae quorum sensing response is mediated by hfq-dependent sRNA/mRNA base pairing interactions*, Molecular Microbiology, 80 (2011), p. 13811394.

[9] A. Barrios, V. Covo, L. Medina, M. Vives-Florez, and L. Achenie, *Quorum quenching analysis in pseudomonas aeruginosa and escherichia coli: network topology and inhibition mechanism effect on the optimized inhibitor dose*, Bioprocess and Biosystems Engineering, 32 (2009), pp. 545–556.

[10] M. Bejerano-Sagie and K. B. Xavier, *The role of small RNAs in quorum sensing*, Current Opinion in Microbiology, 10 (2007), pp. 189–198.

[11] C. Belta, J. Schug, T. Dang, V. Kumar, G. Pappas, H. Rubin, and P. Dunlap, *Stability and reachability analysis of a hybrid model of luminescence in the marine bacterium vibrio fischeri*, in Decision and Control, 2001. Proceedings of the 40th IEEE Conference on, vol. 1, 2001, pp. 869–874 vol.1.

[12] R. G. Brennan and T. M. Link, *Hfq structure, function and ligand binding*, Current Opinion in Microbiology, 10 (2007), pp. 125–133.

[13] K. K. Carter, J. J. Valdes, and W. E. Bentley, *Pathway engineering via quorum sensing and sRNA riboregulatorsInterconnected networks and controllers*, Metabolic Engineering, 14 (2012), pp. 281–288.

[14] F. P. Casey, D. Baird, Q. Feng, R. N. Gutenkunst, J. J. Waterfall, C. R. Myers, K. S. Brown, R. A. Cerione, and J. P. Sethna, *Optimal experimental design in an epidermal growth factor receptor signalling and down-regulation model*, IET systems biology, 1 (2007), pp. 190–202. PMID: 17591178.

[15] J. Chatterjee, C. M. Miyamoto, and E. A. Meighen, *Autoregulation of* luxR*: the* Vibrio harveyi lux*-operon activator functions as a repressor*, Molecular Microbiology, 20 (1996), pp. 415–425.

[16] D. Chopp, M. Kirisits, B. Moran, and M. Parsek, *A mathematical model of quorum sensing in a growing bacterial biofilm*, Journal of Industrial Microbiology & Biotechnology, 29 (2002), p. 339.

[17] D. Chopp, M. Kirisits, B. Moran, and M. Parsek, *The dependence of quorum sensing on the depth of a growing biofilm*, Bulletin of Mathematical Biology, 65 (2003), pp. 1053–1079.

[18] M. W. Covert, E. M. Knight, J. L. Reed, M. J. Herrgard, and B. O. Palsson, *Integrating high-throughput and computational data elucidates bacterial networks*, Nature, 429 (2004), pp. 92–96.

[19] R. Czajkowski and S. Jafra, *Quenching of acyl-homoserine lactone-dependent quorum sensing by enzymatic disruption of signal molecules*, Acta Biochimica Polonica, 56 (2009), pp. 1–16.

[20] T. R. de Kievit and B. H. Iglewski, *Bacterial quorum sensing in pathogenic relationships*, Infect. Immun., 68 (2000), pp. 4839–4849.

[21] T. Defoirdt, N. Boon, P. Sorgeloos, W. Verstraete, and P. Bossier, *Quorum sensing and quorum quenching in vibrio harveyi: lessons learned from in vivo work*, ISME J, 2 (2007), pp. 19–26.

[22] T. Defoirdt, P. Bossier, P. Sorgeloos, and W. Verstraete, *The impact of mutations in the quorum sensing systems of* Aeromonas hydrophila, Vibrio anguillarum and Vibrio harveyi *on their virulence towards gnotobiotically cultured* Artemia franciscana, Environmental Microbiology, 7 (2005), pp. 1239–1247.

[23] S. P. Diggle, A. Gardner, S. A. West, and A. S. Griffin, *Evolutionary theory of bacterial quorum sensing: when is a signal not a signal?*, Philosophical Transactions of the Royal Society B: Biological Sciences, 362 (2007), pp. 1241–1249.

[24] J. Dockery and J. Keener, *A mathematical model for quorum sensing in pseudomonas aeruginosa*, Bulletin of Mathematical Biology, 63 (2001), pp. 95–116.

[25] Y. Dong and L. Zhang, *Quorum sensing and quorum-quenching enzymes*, Journal of microbiology, 43 (2005), pp. 101–109.

[26] P. V. Dunlap, *Regulation of luminescence by cyclic AMP in cya-like and crp-like mutants of vibrio fischeri.*, J. Bacteriol., 171 (1989), pp. 1199–1202.

[27] L. Eberl, *Quorum sensing in the genus burkholderia*, International Journal of Medical Microbiology, 296 (2006), pp. 103–110.

[28] J. Elf, J. Paulsson, O. G. Berg, and M. Ehrenberg, *Near-Critical phenomena in intracellular metabolite pools*, Biophysical Journal, 84 (2003), pp. 154–170.

[29] T. Ellis, X. Wang, and J. J. Collins, *Diversity-based, model-guided construction of synthetic gene networks with predicted functions*, Nature Biotechnology, 27 (2009), pp. 465–471.

[30] M. G. Fagerlind, P. Nilsson, M. Harln, S. Karlsson, S. A. Rice, and S. Kjelleberg, *Modeling the effect of acylated homoserine lactone antagonists in pseudomonas aeruginosa*, Biosystems, 80 (2005), pp. 201–213.

[31] M. G. Fagerlind, S. A. Rice, P. Nilsson, M. Harlén, S. James, T. Charlton, and S. Kjelleberg, *The role of regulators in the expression of quorum-sensing signals in Pseudomonas aeruginosa*, Journal of Molecular Microbiology and Biotechnology, 6 (2003), pp. 88–100.

[32] S. M. Faruque and G. B. Nair, *Vibrio Cholerae: Genomics and Molecular Biology*, Caister Academic Press, Norfolk, UK, 2008.

[33] A. T. Fenley, S. K. Banik, and R. V. Kulkarni, *Computational modeling of differences in the quorum sensing induced luminescence phenotypes of vibrio harveyi and vibrio cholerae*, Journal of Theoretical Biology, 274 (2011), pp. 145–153.

[34] K. S. Frhlich and J. Vogel, *Activation of gene expression by small RNA*, Current Opinion in Microbiology, 12 (2009), pp. 674–682.

[35] C. Fuqua, S. C. Winans, and E. P. Greenberg, *CENSUS AND CONSENSUS IN BACTERIAL ECOSYSTEMS: the LuxR-LuxI family of Quorum-Sensing transcriptional regulators*, Annual Review of Microbiology, 50 (1996), pp. 727–751.

[36] T. A. Geissmann and D. Touati, *Hfq, a new chaperoning role: binding to messenger RNA determines access for small RNA regulator*, EMBO J, 23 (2004), pp. 396–405.

[37] B. Gomez-Gil, S. Soto-Rodriguez, A. Garcia-Gasca, A. Roque, R. Vazquez-Juarez, F. L. Thompson, and J. Swings, *Molecular identification of vibrio harveyi-related isolates associated with diseased aquatic organisms*, Microbiology, 150 (2004), p. 1769.

[38] A. Goryachev, D. Toh, and T. Lee, *Systems analysis of a quorum sensing network: Design constraints imposed by the functional requirements, network topology and kinetic constants*, Biosystems, 83 (2006), pp. 178–187.

[39] A. B. Goryachev, D. Toh, K. B. Wee, T. Lee, H. Zhang, and L. Zhang, *Transition to quorum sensing in an agrobacterium population: A stochastic model*, PLoS Comput Biol, 1 (2005), p. e37.

[40] E. L. Haseltine and F. H. Arnold, *Implications of rewiring bacterial quorum sensing*, Appl. Environ. Microbiol., 74 (2008), pp. 437–445.

[41] J. M. Henke and B. L. Bassler, *Bacterial social engagements*, Trends in Cell Biology, 14 (2004), pp. 648–656.

[42] ——, *Three parallel Quorum-Sensing systems regulate gene expression in vibrio harveyi*, J. Bacteriol., 186 (2004), pp. 6902–6914.

[43] D. Hong, W. M. Saidel, S. Man, and J. V. Martin, *Extracellular noise-induced stochastic synchronization in heterogeneous quorum sensing network*, Journal of Theoretical Biology, 245 (2007), pp. 726–736.

[44] G. A. Hunter and J. P. Keener, *Mechanisms underlying the additive and redundant qrr phenotypes in vibrio harveyi and vibrio cholerae*, Journal of Theoretical Biology, 340 (2014), pp. 38–49.

[45] G. A. M. Hunter, F. Guevara Vasquez, and J. P. Keener, *A mathematical model and quantitative comparison of the small RNA circuit in the vibrio harveyi and vibrio cholerae quorum sensing systems*, Physical Biology, 10 (2013), p. 046007.

[46] A. Ishihama, *Modulation of the nucleoid, the transcription apparatus, and the translation machinery in bacteria for stationary phase survival*, Genes to Cells, 4 (1999), p. 135143.

[47] S. James, P. Nilsson, G. James, S. Kjelleberg, and T. Fagerstrm, *Luminescence control in the marine bacterium vibrio fischeri: an analysis of the dynamics of lux regulation*, Journal of Molecular Biology, 296 (2000), pp. 1127–1137.

[48] L. Jayasree, P. Janakiram, and R. Madhavi, *Characterization of* Vibrio spp. *associated with diseased shrimp from culture ponds of andhra pradesh (India)*, Journal of the World Aquaculture Society, 37 (2006), pp. 523–532.

[49] S. Jian-Wei, *Dynamics and mechanism of a quorum sensing network regulated by small RNAs in vibrio harveyi*, Communications in Theoretical Physics, 55 (2011), p. 465.

[50] J. R. Karr, J. C. Sanghvi, D. N. Macklin, M. V. Gutschow, J. M. Jacobs, B. Bolival, N. Assad-Garcia, J. I. Glass, and M. W. Covert, *A whole-cell computational model predicts phenotype from genotype*, Cell, 150 (2012), pp. 389–401.

[51] L. Keller and M. G. Surette, *Communication in bacteria: an ecological and evolutionary perspective.*, Nature Reviews Microbiology, 4 (2006), pp. 249–258.

[52] J. C. v. Kessel, S. T. Rutherford, Y. Shao, A. F. Utria, and B. L. Bassler, *Individual and combined roles of the master regulators AphA and LuxR in control of the vibrio harveyi quorum-sensing regulon*, Journal of Bacteriology, 195 (2013), pp. 436–443.

[53] R. M. Kumar and J. J. Collins, *Cellular signal processing: Out of one, many*, Molecular Cell, 45 (2012), pp. 143–144.

[54] S. Kwon, *Single-molecule fluorescence in situ hybridization: Quantitative imaging of single RNA molecules*, BMB reports, 46 (2013), pp. 65–72.

[55] D. H. Lenz and B. L. Bassler, *The small nucleoid protein fis is involved in Vibrio cholerae quorum sensing*, Molecular Microbiology, 63 (2007), pp. 859–871.

[56] D. H. Lenz, M. B. Miller, J. Zhu, R. V. Kulkarni, and B. L. Bassler, *CsrA and three redundant small RNAs regulate quorum sensing in* Vibrio cholerae, Molecular Microbiology, 58 (2005), pp. 1186–1202.

[57] D. H. Lenz, K. C. Mok, B. N. Lilley, R. V. Kulkarni, N. S. Wingreen, and B. L. Bassler, *The small RNA chaperone hfq and multiple small RNAs control quorum sensing in vibrio harveyi and vibrio cholerae*, Cell, 118 (2004), pp. 69–82.

[58] E. Levine and T. Hwa, *Small RNAs establish gene expression thresholds*, Current opinion in microbiology, 11 (2008), pp. 574–579.

[59] B. Lewin, *Genes VII*, Oxford University Press, Great Clarendon St, Oxford OX2 6DP, 2000.

[60] X. Li-Ping, M. Yu-Qiang, and T. Lei-Han, *Synergistic effect of auto-activation and small RNA regulation on gene expression*, Chinese Physics Letters, 27 (2010), p. 098701.

[61] W. Lin, G. Kovacikova, and K. Skorupski, *Requirements for vibrio cholerae HapR binding and transcriptional repression at the hapR promoter are distinct from those at the aphA promoter*, J. Bacteriol., 187 (2005), pp. 3013–3019.

[62] T. Long, K. C. Tu, Y. Wang, P. Mehta, N. P. Ong, B. L. Bassler, and N. S. Wingreen, *Quantifying the integration of Quorum-Sensing signals with Single-Cell resolution*, PLoS Biol, 7 (2009), p. e1000068.

[63] M. Manefield, L. Harris, S. A. Rice, R. de Nys, and S. Kjelleberg, *Inhibition of luminescence and virulence in the black tiger prawn (Penaeus monodon) pathogen vibrio harveyi by intercellular signal antagonists*, Appl. Environ. Microbiol., 66 (2000), pp. 2079–2084.

[64] M. Manefield, T. B. Rasmussen, M. Henzter, J. B. Andersen, P. Steinberg, S. Kjelleberg, and M. Givskov, *Halogenated furanones inhibit quorum sensing through accelerated LuxR turnover*, Microbiology, 148 (2002), pp. 1119–1127.

[65] P. Mehta, S. Goyal, T. Long, B. L. Bassler, and N. S. Wingreen, *Information processing and signal integration in bacterial quorum sensing*, Mol Syst Biol, 5 (2009).

[66] P. Mehta, S. Goyal, and N. S. Wingreen, *A quantitative comparison of sRNA-based and protein-based gene regulation*, Molecular Systems Biology, 4 (2008), pp. 221–221.

[67] P. Mehta, R. Mukhopadhyay, and N. S. Wingreen, *Exponential sensitivity of noise-driven switching in genetic networks*, Physical Biology, 5 (2008), p. 026005.

[68] D. L. Milton, *Quorum sensing in vibrios: Complexity for diversification*, International Journal of Medical Microbiology, 296 (2006), pp. 61–71.

[69] N. Mitarai, A. M. C. Andersson, S. Krishna, S. Semsey, and K. Sneppen, *Efficient degradation and expression prioritization with small RNAs*, Physical Biology, 4 (2007), pp. 164–171.

[70] K. C. Mok, N. S. Wingreen, and B. L. Bassler, *Vibrio harveyi quorum sensing: a coincidence detector for two autoinducers controls gene expression*, EMBO J, 22 (2003), pp. 870–881.

[71] N. Musa, L. S. Wei, and W. Wee, *Phenotypic and genotypic characteristics of vibrio harveyi isolated from black tiger shimp (Penaeus monodon)*, World Applied Sciences Journal, 3 (2008), p. 885902.

[72] D. Nacci, M. Huber, D. Champlin, S. Jayaraman, S. Cohen, E. Gauger, A. Fong, and M. Gomezchiarri, *Evolution of tolerance to PCBs and susceptibility to a bacterial pathogen (Vibrio harveyi) in atlantic killifish (Fundulus heteroclitus) from new bedford (MA, USA) harbor*, Environmental Pollution, 157 (2009), pp. 857–864.

[73] C. D. Nadell and B. L. Bassler, *A fitness trade-off between local competition and dispersal in vibrio cholerae biofilms*, Proceedings of the National Academy of Sciences of the United States of America, 108 (2011), pp. 14181–14185. PMID: 21825170 PMCID: PMC3161532.

[74] K. H. Nealson, T. Platt, and J. W. Hastings, *Cellular control of the synthesis and activity of the bacterial luminescent system*, Journal of Bacteriology, 104 (1970), pp. 313–322.

[75] M. B. Neiditch, M. J. Federle, S. T. Miller, B. L. Bassler, and F. M. Hughson, *Regulation of LuxPQ receptor activity by the Quorum-Sensing signal autoinducer-2*, Molecular Cell, 18 (2005), pp. 507–518.

[76] M. B. Neiditch, M. J. Federle, A. J. Pompeani, R. C. Kelly, D. L. Swem, P. D. Jeffrey, B. L. Bassler, and F. M. Hughson, *Ligand-Induced asymmetry in histidine sensor kinase complex regulates quorum sensing*, Cell, 126 (2006), pp. 1095–1108.

[77] W. Ng and B. L. Bassler, *Bacterial Quorum-Sensing network architectures*, Annual Review of Genetics, 43 (2009), pp. 197–222.

[78] W.-L. Ng, L. J. Perez, Y. Wei, C. Kraml, M. F. Semmelhack, and B. L. Bassler, *Signal production and detection specificity in vibrio CqsA/CqsS quorum-sensing systems*, Molecular Microbiology, 79 (2011), pp. 1407–1417. PMID: 21219472 PMCID: PMC3285556.

[79] M. R. Parsek and E. P. Greenberg, *Acyl-homoserine lactone quorum sensing in gram-negative bacteria: A signaling mechanism involved in associations with higher organisms*, Proceedings of the National Academy of Sciences of the United States of America, 97 (2000), pp. 8789–8793.

[80] M. H. Pontes, M. Babst, R. Lochhead, K. Oakeson, K. Smith, and C. Dale, *Quorum sensing primes the oxidative stress response in the insect endosymbiont, sodalis glossinidius*, PLoS ONE, 3 (2008).

[81] J. L. Ramos, M. Martinez-Bueno, A. J. Molina-Henares, W. Teran, K. Watanabe, X. Zhang, M. T. Gallegos, R. Brennan, and R. Tobes, *The TetR family of transcriptional repressors*, Microbiol. Mol. Biol. Rev., 69 (2005), pp. 326–356.

[82] V. A. Rhodius and S. J. Busby, *Positive activation of gene expression*, Current Opinion in Microbiology, 1 (1998), pp. 152–159.

[83] E. Ross, I. Taub, C. Doona, F. Feeherry, and K. Kustin, *The mathematical properties of the quasi-chemical model for microorganism growth-death kinetics in foods*, International Journal of Food Microbiology, 99 (2005), pp. 157–171.

[84] E. G. Ruby and K. H. Nealson, *Symbiotic association of* Photobacterium fischeri *with the marine luminous fish* Monocentris japonica*; a model of symbiosis based on bacterial studies*, The Biological Bulletin, 151 (1976), pp. 574–586.

[85] S. T. Rutherford, J. C. van Kessel, Y. Shao, and B. L. Bassler, *AphA and LuxR/HapR reciprocally control quorum sensing in vibrios*, Genes & Development, 25 (2011), pp. 397–408. PMID: 21325136 PMCID: PMC3042162.

[86] A. Safa, J. Sultana, P. D. Cam, J. C. Mwansa, and R. Y. Kong, Vibrio cholerae *O1 hybrid El Tor strains, Asia and Africa*, Emerging Infectious Diseases, 14 (2008), p. 987.

[87] M. Schuster, C. P. Lostroh, T. Ogi, and E. P. Greenberg, *Identification, timing, and signal specificity of* Pseudomonas aeruginosa *Quorum-Controlled genes: a transcriptome analysis*, J. Bacteriol., 185 (2003), pp. 2066–2079.

[88] M. Schuster, D. J. Sexton, S. P. Diggle, and E. P. Greenberg, *Acyl-homoserine lactone quorum sensing: From evolution to application*, Annual Review of Microbiology, 67 (2013), p. null. PMID: 23682605.

[89] S. A. Shabalina, A. Y. Ogurtsov, V. A. Kondrashov, and A. S. Kondrashov, *Selective constraint in intergenic regions of human and mouse genomes*, Trends in Genetics, 17 (2001), pp. 373–376.

[90] G. S. Shadel and T. O. Baldwin, *The vibrio fischeri LuxR protein is capable of bidirectional stimulation of transcription and both positive and negative regulation of the luxR gene.*, Journal of Bacteriology, 173 (1991), pp. 568–574.

[91] Y. Shao and B. L. Bassler, *Quorum-sensing non-coding small RNAs use unique pairing regions to differentially control mRNA targets*, Molecular Microbiology, 83 (2012), pp. 599–611. PMID: 22229925 PMCID: PMC3262071.

[92] C. Smith, H. Song, and L. You, *Signal discrimination by differential regulation of protein stability in quorum sensing*, Journal of Molecular Biology, 382 (2008), pp. 1290–1297.

[93] P. S. Stewart, *Diffusion in biofilms*, J. Bacteriol., 185 (2003), pp. 1485–1491.

[94] S. L. Svenningsen, K. C. Tu, and B. L. Bassler, *Gene dosage compensation calibrates four regulatory RNAs to control vibrio cholerae quorum sensing*, The EMBO Journal, 28 (2009), pp. 429–439.

[95] S. L. Svenningsen, C. M. Waters, and B. L. Bassler, *A negative feedback loop involving small RNAs accelerates vibrio choleraes transition out of quorum-sensing mode*, Genes & Development, 22 (2008), pp. 226–238.

[96] L. R. Swem, D. L. Swem, N. S. Wingreen, and B. L. Bassler, *Deducing receptor signaling parameters from in vivo analysis: LuxN/AI-1 quorum sensing in vibrio harveyi*, Cell, 134 (2008), pp. 461–473.

[97] Y. Tanouchi, D. Tu, J. Kim, and L. You, *Noise reduction by diffusional dissipation in a minimal quorum sensing motif*, PLoS Comput Biol, 4 (2008), p. e1000167.

[98] A. F. Taylor, M. R. Tinsley, F. Wang, Z. Huang, and K. Showalter, *Dynamical quorum sensing and synchronization in large populations of chemical oscillators*, Science, 323 (2009), pp. 614–617.

[99] S.-W. Teng, J. N. Schaffer, K. C. Tu, P. Mehta, W. Lu, N. P. Ong, B. L. Bassler, and N. S. Wingreen, *Active regulation of receptor ratios controls integration of quorum-sensing signals in vibrio harveyi*, Molecular Systems Biology, 7 (2011), p. 491. PMID: 21613980 PMCID: PMC3130561.

[100] I. Thiele, N. Swainston, R. M. T. Fleming, A. Hoppe, S. Sahoo, M. K. Aurich, H. Haraldsdottir, M. L. Mo, O. Rolfsson, M. D. Stobbe, S. G. Thorleifsson, R. Agren, C. Blling, S. Bordel, A. K. Chavali, P. Dobson, W. B. Dunn, L. Endler, D. Hala, M. Hucka, D. Hull, D. Jameson, N. Jamshidi, J. J. Jonsson, N. Juty, S. Keating, I. Nookaew, N. Le Novre, N. Malys, A. Mazein, J. A. Papin, N. D. Price, E. S. Sr, M. I. Sigurdsson, E. Simeonidis, N. Sonnenschein, K. Smallbone, A. Sorokin, J. H. G. M. van Beek, D. Weichart, I. Goryanin, J. Nielsen, H. V. Westerhoff, D. B. Kell, P. Mendes, and B. . Palsson, *A community-driven global reconstruction of human metabolism*, Nature Biotechnology, advance online publication (2013).

[101] F. L. Thompson, B. Austin, and J. Swings, eds., *The Biology of Vibrios*, ASM Press, 1752 N Street NW, Washington, DC 20036-2804, 2006.

[102] K. C. Tu and B. L. Bassler, *Multiple small RNAs act additively to integrate sensory information and control quorum sensing in vibrio harveyi*, Genes & Development, 21 (2007), pp. 221–233.

[103] K. C. Tu, T. Long, S. L. Svenningsen, N. S. Wingreen, and B. L. Bassler, *Negative feedback loops involving small regulatory RNAs precisely control the vibrio harveyi Quorum-Sensing response*, Molecular Cell, 37 (2010), pp. 567–579.

[104] K. C. Tu, C. M. Waters, S. L. Svenningsen, and B. L. Bassler, *A small-RNA-mediated negative feedback loop controls quorum-sensing dynamics in vibrio harveyi*, Molecular Microbiology, 70 (2008), pp. 896–907.

[105] V. Venturi, *Regulation of quorum sensing in* Pseudomonas, FEMS Microbiology Reviews, 30 (2006), pp. 274–291.

[106] V. Venturi, A. Friscina, I. Bertani, G. Devescovi, and C. Aguilar, *Quorum sensing in the burkholderia cepacia complex*, Research in Microbiology, 155 (2004), pp. 238–244.

[107] K. J. Versyck, K. Bernaerts, A. H. Geeraerd, and J. F. Van Impe, *Introducing optimal experimental design in predictive modeling: A motivating example*, International Journal of Food Microbiology, 51 (1999), pp. 39–51.

[108] L. Vial, M. Groleau, V. Dekimpe, and E. Dexel, *Burkholderia diversity and versatility: an inventory of the extracellular products.*, Journal of microbiology and biotechnology, 17 (2007), pp. 1407–1429.

[109] M. Vidgen, J. Carson, M. Higgins, and L. Owens, *Changes to the phenotypic profile of* Vibrio harveyi *when infected with the* Vibrio harveyi *myovirus-like (VHML) bacteriophage*, Journal of Applied Microbiology, 100 (2006), pp. 481–487.

[110] K. L. Visick, J. Foster, J. Doino, M. McFall-Ngai, and E. G. Ruby, *Vibrio fischeri lux genes play an important role in colonization and development of the host light organ*, J. Bacteriol., 182 (2000), pp. 4578–4586.

[111] J. Vogel and K. Papenfort, *Small non-coding RNAs and the bacterial outer membrane*, Current Opinion in Microbiology, 9 (2006), pp. 605–611.

[112] C. M. Waters and B. L. Bassler, *QUORUM SENSING: Cell-to-Cell communication in bacteria*, Annual Review of Cell and Developmental Biology, 21 (2005), pp. 319–346.

[113] Y. Wei, W.-L. Ng, J. Cong, and B. L. Bassler, *Ligand and antagonist driven regulation of the vibrio cholerae quorum-sensing receptor CqsS*, Molecular Microbiology, 83 (2012), pp. 1095–1108. PMID: 22295878 PMCID: PMC3310172.

[114] Y. Wei, L. J. Perez, W.-L. Ng, M. F. Semmelhack, and B. L. Bassler, *Mechanism of vibrio cholerae autoinducer-1 biosynthesis*, ACS Chemical Biology, 6 (2011), pp. 356–365. PMID: 21197957 PMCID: PMC3077805.

[115] N. A. Whitehead, A. M. Barnard, H. Slater, N. J. Simpson, and G. P. Salmond, *Quorum-sensing in gram-negative bacteria*, FEMS Microbiology Reviews, 25 (2001), pp. 365–404.

[116] J. W. Williams, X. Cui, A. Levchenko, and A. M. Stevens, *Robust and sensitive control of a quorum-sensing circuit by two interlocked feedback loops*, Mol Syst Biol, 4 (2008).

[117] P. Williams and M. Cámara, *Quorum sensing and environmental adaptation in pseudomonas aeruginosa: a tale of regulatory networks and multifunctional signal molecules*, Current Opinion in Microbiology, 12 (2009), pp. 182–191.

[118] P. Williams, K. Winzer, W. C. Chan, and M. Cmara, *Look who's talking: communication and quorum sensing in the bacterial world*, Philosophical Transactions of the Royal Society B: Biological Sciences, 362 (2007), pp. 1119–1134.

[119] T. Wilson and J. W. Hastings, *BIOLUMINESCENCE*, Annual Review of Cell and Developmental Biology, 14 (1998), pp. 197–230.